Inhaltsverzeichnis

1 Wissensfragen

Leitungskodierung

Was bedeutet Qos?
Quality of Service, d.h. Dienstgüteeigenschaften

Ein Rechner hat die IP-Adresse 192.168.20.5/16 (klassenlose Adressie rung). Wie lautet die dazugehoerige Subnetzwerk?
Ersten 16 Bit der IP-Adresse -> 192.168.0.0

Zur Klassifiaktionkation von Kommunikationssystemen soll die Übertragungsart betrachtet werden. Nennen Sie zwei Klassen der Übertragungsrichtung.
simplex,duplex

Was bedeutet P2P? Nennen Sie außerdem die Grundidee, eine Eigen- schaft, ein Anwendungsgebiet und ein konkretes Beispiel von P2P.
Bedeutung und Grundidee:
Peer To Peer Nicht ein großer, zentraler Server mit guter Übetragungsrate, sondern sehr viele kleine Rechner (Peers), die immer nur einen kleinen Teil übertragen. In Summe aber sehr mächtig.

Eigenschaften: Hohe Konnektivität, Overlay-Netz, Hauptteil des Netzverkehrs heutzutage

Anwendung: Musikaustausch, Filmaustausch Beipsiel: Bittorrent, Napster

Was ist die Aufgabe der Sicherungsschicht?
Transfer der Rahmen zwischen Knoten über Links

Beantworten Sie jeweils für Flusskontrolle und Überlastkontrolle: Wer wird vor Überlast geschützt? Wer wird kontrolliert?
Flusskontrolle: Nutzer wird geschützt, Kanal kontrolliert
Überlastkontrolle: Kanal wird geschützt, Nutzer kontrolliert

Welche zwei Kontrollmechanismen gibt es bei TCP und wofür sind sie da (kurze Erklärung)?
Flusskontrolle und Überlastkontrolle. Flusskontrolle schützt den Empfänger vor Überlast und Überlastkontrolle schützt das Netz vor Überlast.

Was ist Modulation? Auf welcher Schicht des Internetprotokollstapels ist Modulation einzuordnen? Nennen Sie eine Modulationsart und hierfür ein Beispiel.
Modulation: Änderung von Paramtern wie der Amplitude, Frequenz oder Phase durch ein modulierendes Signal
Physikalische Schicht Z.B. Analog-Analog-Wandlung: Amplitudenmodulation,
Beispiel: TV-Signal

Unter welchen Bedingungen bietet ALOHA einen höheren Durchsatz als ein CSMA-basiertes Verfahren (grobe ganzzahlige Abschätzung)? Nennen Sie ein Anwendungsbei- spiel.
ALOHA hängt nicht von der Kanalpuffergröße ab, CSMA schon.
Bei hohem a ist deswegen ALOHA besser
sCSMA = 1/(1+6,4a) sALOHA = 1/2e = 0,18
Abschätzung:
$1/(1+6.4a) < 0{,}18$
$1 < 0.18\ (1+6.4a)$
$1 < 0{,}18 + 1{,}152a$
$0.82 < 1.152a$
$a > 0.712$
Anwbsp: Satelitennetz

Wodurch wird bei Berechnung der Prufsumme in TCP und UDP das Schichtenprinzip verletzt?

TCP und UPD sind Transportschicht und damit nicht in der Verbinndungs/-Sicherungschicht. Ausserdem bilden die Protokolle Checksums über informationen im Header die in ihrer Schicht eigentlich noch nicht verfuegbar sein sollten (naehmlich IP-Header-Informationen, Netzwerk aus UDP/TCP, Transport)

??

Ein Computer fordert eine Webseite von einem Server an (GET /index.html; TCP-Port 80). Gehen Sie davon aus, dass bereits alle benötigten Informationen (Netzwerk-Adressen) vorliegen. Zeichnen Sie in Abbildung 2 schematisch den Aufbau des vom Computer zu versendenden Rahmens in dem der HTTP-Request enthalten ist (= drittes Paket des 3- Wege-Handshakes) ein. Markieren Sie eindeutig die einzelnen Header und tragen Sie die wichtigen Informationen ein.

Praeambel
————- IP-Segment (Pseudoheader, Netzwerkschicht)
6 Protokoll TCP (17 ist UDP, 6 TCP)
10.0.1.2 Quell IP
10.0.3.4 Ziel IP
————- TCP-Segment
128 80 SourcePort DestPort
0 Sequence Number
0 ACK number
00000000 0 Flags und AW
0 0 Checksum/ Urg data pnter
GET /index.html DATA

Nennen Sie je einen Vor- und einen Nachteil von Cut-through switching im Vergleich zum Store-and-Forward switching.

\+ : schnelleres Weiterleiten von Daten
\- : fehlerhafte Daten können leicht weitergeleitet werden

Zu welchem Zweck werden digitale Signale vor der Übertragung einer Leitungskodierung unterzogen? Nennen Sie zwei Eigenschaften.

Um die Signale sicher zu übertragen
Eigenschaften: selbsttaktend, gleichstromfrei

Welcher Zusammenhang besteht bei der einfachen Manchesterkodierung zwischen Signal- und Übertragungsrate?

Übertragungssrate = 1/2 * Signalrate
Manchester ist selbsttaktend. Dementsprechend kann man aus dem Signal die Übertragungsrate gewinnen.

Nennen Sie vier weitere Möglichkeiten der Leitungskodierung.

NRZ, NRZI, Bipolar, Differentielle Manchester-Kodierung

Wie kann Host A in einem lokalem Ethernet-Netzwerk (alle Hosts sind über einen Switch miteinander verbunden) die physikalische Adresse (MAC-Adresse) von Host B herausfinden? Geben Sie das zugrunde liegende Protokoll an und beschreiben Sie stichpunktartig dessen Funktionsweise. Auf welcher Schicht ist dieses Protokoll angesiedelt?
Protokoll: ARP
Ebene: Sicherungsschicht.
A mach Brotkast mit IP von B
B sieht seine IP und antwortet A mit seiner MAC

Muss ein Host grundsätzlich die MAC-Adresse seines Kommunikationspartners ken- nen um mit ihm kommunizieren zu können? Begrü unden Sie ihre Antwort.
Nein, Host A könnte einfach immer Brotkasten und host B antworten. A würde damit nie die MAC-Adresse von B kennen und trozdem mit ihm kommunizieren.

Nennen Sie eine klassische Anwendung die üblicherweise mittels UDP kommuniziert. Was ist der Vorteil im Gegensatz zu TCP? Begründen Sie kurz Ihre Antwort.
Videoübertragung (alternativ: Internettelefonie), wenn ein einzelnes Paket/Bild verloren geht, kann das einfach ignoriert werden, UDP toleriert Fehler, TCP verhindert sie aktiv, was aber zu größeren Verzögerungen führen kann.

Nennen Sie eine klassische Anwendung die üblicherweise mittels TCP kommuniziert. Was ist hier der Vorteil im Gegensatz zu UDP? Begründen Sie kurz Ihre Antwort.
TCP stellt sicher, dass keine Pakete verloren gehen und dass alle in der richtigen Reihenfolge ankommen. Es findet zum Beispiel bei SMTP (alternativ: Remote Terminal Access) also E-Mails Anwendung.

Könnten theoretisch die Anwendungen aus den Teilaufgaben a) und b) auch das jeweils andere Übertragungsprotokoll verwenden? Was wäre ein Nachteil bzw. was müsste beachtet werden? Bitte erläutern Sie beide Fälle.
Videoübertragung könnte natürlich auch TCP verwenden, aber es könnte zu Verzögerungen/lags kommen, wenn komplette RTT 's auf fehlende Pakete gewartet werden muss.
Mail mit UDP wäre problematisch. Der Server müsste die Verbinndung zum MUA aktiv halten, solange er ürprüft ob alle Pakete korrekt angekommen sind. Defacto würde man TCP mit UDP nachimplementieren können, was wenig sinnvoll wäre.

Erkären Sie den Begriff Out-of-Band-Control anhand des FTP-Protokolls. Was ist der entscheidende Vorteil?

Steuerbefehle gehen nicht über die normale Datenverbindung, sondern über eine extra aufgebaute Verbinndung.
Vorteil: Steuerung auch während Datenübertragung möglich

Erklären Sie kurz die Funktionsweise des Passive Modes im FTP-Protokoll. Nennen Sie den wesentlichen Vorteil dieser Datenübertragung im Vergleich zum Active Mode ?
- Client baut Verbinndung von Port N (N>1024) zum port 21 des Servers auf (Steuerverbinndung)
- Client sendet Befehl PASV
- Server antwortet mit beliebigen unpriviligierten Port P und wartet an diesem Port auf Verbinndung
- Client baut TCP-Verbindung zum Port P vom Port N+1 auf
Vorteil: Firewall des Clients muss Verbindung von Port N+1 durch den Server nicht explizit zulassen, da im Passiv-Mode der Client die Verbindung aufbaut.

Listen Sie die 5 Schichten des Internet-Protokollstapels von oben nach unten auf.
Anwendungsschicht: SMTP, http
Transportschicht: TCP/UDP
Netzwerkschicht: IP
Sicherungsschicht: CSMA, MAC
Physikalische Schicht: Modulation

Ein Client möchte eine TCP-Verbindung mit einem Server aufbauen.
3-Wege Handshake:
Client sendet SYN-Segment mit SYN-Flag=1, zufälliger initialer CLient-SQN (client_isn), ohne Daten
Server sendet SYNACK Segment mit SYN-FLAG=ACK-Flag=1, zufälliger initialer Server-SQN (server_isn), ACK=client_isn+1, ohne Daten, er legt Puffer und Varriablen an
Client sender Segment mit ACK-Flag=1, SQN=client_isn+1, ACK=server_isn+1 und ggf. Daten; er legt Puffer und Variable an

Wann ist CSMA bzw. ALOHA besser?
Für kleines a ist CSMA besser, für großes a ALOHA besser
CSMA>ALOHA: wenn knoten sich absprechen können müssen, bsp: falls es prioritäten, oder abhängigkeiten der Knoten gibt
ALOHA>CSMA : bei kleinen einfachen Netzwerken, Satellitenübertragung
CSMA: größerer maximaler Durchsatz
ALOHA: größerer Durchsatz für große N

Geben Sie je einen Vorteil und einen Nachteil der Manchester-Codierung gegenuber der NRZ-Codierung an.

Vorteil: Ein zusätzlicher Taktgeber wird nicht benötigt, aus dem Code selbst kann das Taktsignal abgeleitet werden kann.
Ein Nachteil der Manchester-Codierung ist, dass bei der Datenübertragung die benötigte Bandbreite doppelt so hoch ist wie bei der einfachen Binärcodierung (z. B. Non Return to Zero, NRZ). Der Grund dafür liegt darin, dass für die Codierung eines Bits zwei Signale benötigt werden. Die Bitrate (im Fall eines zweiwertigen Signals) ist somit nur halb so groß wie die Baudrate.

Listen Sie drei wichtige Anwendungsschicht-Protokolle auf, die bei der Ubertragung von E-Mails verwendet werden.
POP3, IMAP, SMTP

Wieso sollte eine ASCII-Textdatei beim Ubertragen mittels SMTP kodiert werden.
Einige zeichenketten z.B. CRLF.CRLF sind in Nachrichten nicht erlaubt. Daher müssen Nachrichten codiert werden (üblicherweise Base-64 oder quoted printable)
(Der Server nutzt nämlich CRLF.CRLF zum Erkennen des Endes einer Nachricht)

Listen Sie die 5 Schichten des Internet-Protokollstapels von oben nach unten auf. Geben Sie fur jede Schicht ein Beispiel (Protokoll, Verfahren, oder Anwendung) an. Ordnen Sie ihnen die jeweiligen Bezeichnungen der PDU zu.

Application - HTTP, FTP -PDU: WIKI : Daten ,Skript : APDU
Transport - TCP, UDP - PDU: WIKI : Segment ,Skript : TPDU
Network - IP (Internet Protocol), Routing - PDU: WIKI: Paket, Skript: Paket
Link - Medienzugriff, Sicherung - PDU: WIKI : Frame ,Skript : Rahmen
Physical - binaere Formate, Modulationsverfahren - PDU: WIKI : BIt ,Skript : Bit

Auf welchen Schichten mussen folgende Netzwerkgeräte jeweils mindestens arbeiten?
Hubs: Physical [vergleichbar mit repeater]
Router: Netzwerkschicht
Repeater: Physikalische Schicht
Switches: Verbindung/Link/Sicherungsschicht

Welchen Effekt stellt die Kanalpuffergröße a dar?
Die maximale Menge an Daten die während der Übertragung im Übertragungsmedium vorliegen kann.

Nennen Sie ein echtzeitfähiges Medienzugriffsverfahren.
Feste Kanalaufteilung (FDMA; TDMA, CDMA)
Großer Vorteil ist, dass das Knotenpaar einen festen Kanal hat, welcher bei echtzeitfähigen Anwendungen (Sprache, Video) dauernd genutzt wird
Token-Ring, immer nur eine Station sendet das Token an die naechste und darf das To-

ken nur fuer eine feste Zeit halten, wenn man dann eine feste Anzahl von Stationen hat kann man sicher sein dass man das Token nach einer bestimmten Zeit bekommt. Und die Tatsache dass es nur ein Token gibt stellt sicher dass es keine kollisionen geben kann.

Kann der Computer in Tabelle 1 direkt mit dem Webserver (auf Netzwerkschicht) kom- munizieren oder benötigt er hierfür einen Router? Nennen Sie die Subnetzadresse in dem sich der Computer befindet. Wie viele Teilnehmer kann es maximal enthalten? (Achtung: alle Host-Bits zu 0 oder zu 1 haben Sonderrollen.)

Router	IP:	10.0.1.1/22
	MAC:	00:00:00:00:00:F1
Computer	IP:	10.0.1.2/22
	MAC:	00:00:00:00:00:F2
Webserver	IP:	10.0.3.4/22
	MAC:	00:00:00:00:00:F3

Tabelle 1: Netzwerkinformationen

PC : 0000 0010.0000 0000.0000 0001.0000 0010
WEB: 0000 0010.0000 0000.0000 0011.0000 0100
22 Bit von Links abtrennen
-> PC: 0000 0010.0000 0000.0000 00 (10.0.0.0)
-> WEB: 0000 0010.0000 0000.0000 00 (10.0.0.0)
PC = WEB -> Kein Router wird benötigt
32 Bit - 22 Bit = 10 Bit
Subnetzwerkadressen: $2^{10} - 2$

Ein Rechner hat die IP-Adresse 192.168.20.9/20 (klassenlose Adressierung). Wie lautet die dazugehörige Subnetzadresse?
IP Adresse: 192.168.20.9/20
20 Steht fuer die Subnetzmaske, und zwar genauer fuer die Anzahl an Bits mit Wert '1'
20 ist also als 1111 1111 . 1111 1111 . 1111 0000 . 0000 0000
oder auch als 255.255.240.0 zu verstehen
Die Subnetzadresse erhalten wir indem wir ein AND (KEINE Addition) auf Maske und IP-Adresse anwenden also
interessant ist hier nur das vorletzte Feld, die ersten beiden behalten ihren Wert da im-

mer mit ’1” Verundet wird, das letzte wird zu ’0’ da immer mit ’0’ Verundent wird
3tes Feld:
Maske: 1111 0000
Adresse: 0001 0100
Subnetz: 0001 0000 (=16), das bedeutet die gesamte Subnetzadresse ist 192.168.16.0

??

Begrunden Sie, wieso TCP bei der Überlastkontrolle konservativer auf das Ende eines Timeouts reagiert als auf drei duplizierte ACKs

Aus Wikipedia: der Sender bemerkt die duplizierten Bestätigungen, und nach dem dritten Duplikat sendet er sofort, vor Ablauf des Timers, das verlorene Paket erneut. Weil nicht auf den Ablauf eines Timers gewartet werden muss, heißt das Prinzip Fast Retransmit. Die Dup-Acks sind auch Hinweise darauf, dass zwar
ein Paketverlust stattfand, aber doch die folgenden Pakete angekommen sind. Deshalb wird das Sendefenster nur halbiert und nicht wie beim Timeout wieder mit Slow-Start begonnen. Zusätzlich kann das Sendefenster noch um die Anzahl der Dup-Acks erhöht werden, denn jedes steht für ein weiteres Paket,
welches den Empfänger erreicht hat, wenn auch außer der Reihe. Da dadurch nach dem Fehler schneller wieder die volle Sendeleistung erreicht wird, nennt man das Prinzip Fast-Recovery.
Ich hab das jetzt so verstanden das das Konservative am Timeout der Slow-Start ist, während bei drei Dup-Acks lediglich halbiert wird. Korrigiert mich wenn ich daneben liege.?
Drei doppelte Acks heißt halt, dass was verloren gegangen ist, aber andere Pakete und auch die doppelten Acks noch ankamen, also scheint das Netz nur leicht überlastet zu sein. Wenn die Stoppuhr komplett abläuft scheint das Netz dagegen total überlastet zu sein,
deswegen fängt man dann wieder mit Slow-Start von vorne an.

Gemäß den Basic Encoding Rules (BER) erhalten Sie folgenden Datenstrom in hexadezimaler Darstellung: 02 02 01 01. Welche Information wurde Ihnen gesendet? Dekodierten Sie die erhaltene Information.

Ein Integer [02]
der Länge 2 [02]
mit Wert: 0x0101 (= 257)

4 Verzögerungszeiten in paketvermittelten Netzen

- Ausbreitungsverzögerung
- Warteschlangenverzögerung
- Bearbeitungsverzögerung
- Übertragungsverzögerung

Paketverlust Beispiele

- Warteschlange voll

- Physikalische Probleme

Wesentlichen Grund fur die minimale Framegröße von 64 Byte bei Ethernet
Damit soll verhindert werden, dass eine Station die Übertragung eines kurzen Rahmens beendet, bevor sein erstes Bit überhaupt das andere Ende des Kabels erreicht, wo er dann vielleicht mit einem anderen Frame kollidiert.

Wofür gibt es eine maximale Grenze der Größe eines Frames?
Damit die Kollisionserkennung funktioniert.

Wieso wird bei Stop-And-Wait eine bereits versendete Bestätigung (ACK) zwischengespeichert?
Wegen möglichem ACK-Verlust/ACK-Verzögerung
Alternativ: Damit immer das ACK für das letzte erfolgreiche Paket geschickt werden kann

Nennen Sie Modifikationen von IPv6 gegenuber von IPv4, die es dem Router ermöglichen Pakete schneller zu verarbeiten
- Network Address Translation (NAT) wird mit IPv6 weitgehend vermieden
- Adressknappheit vorgebaut
- Header mit fester Länge (für schnelles Weiterleiten)
- keine Fragmentierung (wird einfach verworfen falls > MTU)
- keine Prüfsumme (Fehlererkennung in höheren Schichten)
- zusätzliche Optionen außerhalb als nächster Header
- zustandslose Autokonfiguration
- Dienstgüte (neben DS und ECN auch Flow-Label)

Weitere Eigenschaften von IPv6
- Abgespeckter Header
- Prioriät von Paketen

Warum wird fur drahtlose Kommunikation in der Regel CSMA/CA (und nicht CSMA/CD) verwendet?
CD steht für collision detection, CA steht für collison avoidance, bei drahtloser Kommunikation kann man collision nicht feststellen (detect), sondern wartet wenn was nicht klappt (=collision avoidance)
→ Kollisionserkennung würde 2. Antenne benötigen, die während des Sendens empfängt
→ schwieriger, teurer

Vor- und Nachteil fur eine statische Kanalaufteilung statt eines zufallsbasierten Zugriffsverfahren.
Nachteil: ineffizient, weil die Daten nur immer mal wieder auf den Kanälen gesendet werden → ineffiziente Auslastung
Vorteil: Keine Kollisionen, echtzeitfaehig

Werden ICMP-Nachrichten zuverlässig ausgeliefert?
So zuverlässig, wie eben die IP Übertragung.
Dient außerdem zur Fehlersignalisierung, muss also zuverlässig sein.

Ein IP-Paket trägt ein TCP-Segment in sich und wird von einem NAT-Router aus dem lokalen Netzwerk in das Internet weitergeleitet. Nennen Sie auf TCP/IP Ebene alle Header-Felder, die der Router fur eine korrekte Weiterleitung verändern muss.
Quelladresse und -Port. Wir senden schliesslich. Beim Empfangen waere es Zieladresse und Port.

Stellen Sie Go-Back-N und Selective-Repeat gegenuber, indem Sie je einen Vor- und einen Nachteil des Verfahrens nennen. Wie lässt sich TCP mit SACK und TCP ohne SACK in diese Verfahren einordnen. Begrunden Sie Ihre Antwort.
Vorteile Go-Back-N:
- kumultative ACKs gleichen ACK-Verluste und Verspätungen schnell aus ohne dass die Pakete erneut gesendet werden müssen
- der Sender benöetigt nur einen Timer
- der Empfaenger benötigt keinen Puffer
- Sender und Empfänger können einfacher realisiert werden, weil keine Lücken in den Fenstern beachtet werden müssen
Vorteile Selective-Repeat:
- weniger Wiederholungen von Sendungen, weil wirklich nur fehlerhafte oder verlorengegangene Pakete erneut gesendet werden (Nachteil von Go-Back-N)
Nachteile Selective Repeat:
- Empfaenger benötigt Puffer
- generell schwieriger zu realisieren
SACK = selective ACK
SACK erhalten (und kumulative ACK) = selective Repeat
keine SACK erhalten (und kein kumulative ACK) = full Go-Back-N

Out-of-Band-Control anhand des FTP-Protokolls
verschiedene TCP-verbindungen für Datenübertragung und Steuerung/Befehle
vorteil: kann Befehle auch bei Dateiübertragung annehmen

Vor- und einen Nachteil des Pseudo-Headers bei UDP
+ Auch Bitfehler in IP-Header können erkannt werden
- Verletzung des Schichtenprinzips

Zu welchem Zweck werden digitale Signale vor der Ubertragung einer Leitungskodierung unterzogen?
Selbsttaktung

Bandbreitenbedarf
Gleichtstromfreiheit

2 Rechnen

2.1 RTT (dProp)

2.1.1

3 POP3-Leistungsanalyse (29 Punkte)

Ein Programm auf Host C holt die erste E-Mail aus dem Postfach von Host S ab. Dazu baut das Programm eine POP3-Verbindung auf (gemäß Spezifikation), ruft die erste E-Mail im Postfach ab und beendet die Verbindung anschließend wieder. Der für die Authentifizierung benötigte Benutzername ist: *user*. Das Passwort lautet: *geheim*.

Folgende Annahmen treffen immer zu: Es kann davon ausgegangen werden, dass sich immer mindestens eine E-Mail auf dem Server befindet (kein *list* Befehl notwendig). Bei der Übertragung treten keine Fehler auf. Die Verbindung zwischen Host C und Host D ist nicht symmetrisch (ähnlich wie zu einer DSL-Verbindung). In beiden Hosts treten keine Verzögerungen auf. Weiterhin gelten die Konstanten gemäß Tabelle 1.

Ausbreitungsverzögerung:	$d_{prop,SC} = 10\,\text{ms}$ (Host S $\rightarrow$ Host C) $d_{prop,CS} = 100\,\text{ms}$ (Host C $\rightarrow$ Host S)
Raten:	$R_{SC} = 10\ \frac{Mbit}{s}$ (Host S $\rightarrow$ Host C) $R_{CS} = 1\ \frac{Mbit}{s}$ (Host C $\rightarrow$ Host S)
Paketlänge:	$L = 1500\ Byte\ (= MSS)$

Tabelle 1: Konstanten

Gibt es auf Grund der asymmetrischen Leitung ausgehend von Host C bzw. von Host S verschiedene Round Trip Times (RTT)? Begrüunden Sie Ihre Antwort und berechnen Sie die RTT.
Antwort: Nein gibt es nicht.
RTT = dpropSC+dpropCS = 110 ms

2.1.2

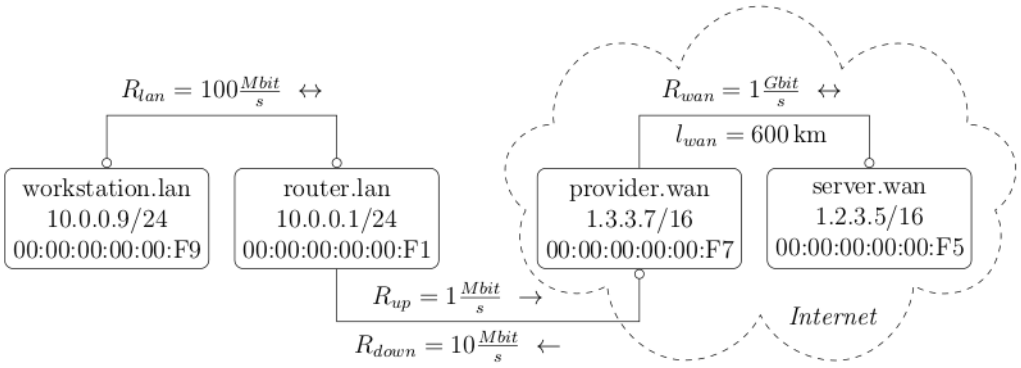


Abbildung 1: Netzwerkkonfiguration

Szenario

Das hier gezeigte Szenario, sowie die hier gelisteten Konstanten gelten für alle folgenden Aufgaben.

Gegeben sei eine Netzwerkkonfiguration gemäß Abbildung 1. Sie besteht aus einem Local Area Network (LAN) und aus einem Wide Area Network (WAN). Beide Netzwerke sind mittels einer asymmetrischen Digital Subscriber Line (DSL) verbunden. Der Router *router.lan* betreibt Paket Weiterleitung und eine Domain Name System (DNS). Jedes Rechteck bildet einen Teilnehmer ab. In der ersten Zeile steht der Domain Name des Teilnehmers. In der zweiten Zeile seine IP und in der dritten Zeile seine Medium Access Control (MAC) Adresse. Die WAN-Informationen des Router Interfaces *router.lan* sowie die Informationen für das zweite Interface von der Gegenstelle *provider.wan* sind nicht gelistet, da sie für diese Aufgabe keine Relevanz haben. (Die gezeigten Informationen gelten für den Link mit Punkt.) An den Links sind jeweils die Verbindungsgeschwindigkeiten in die jeweilige Richtung (Pfeile) annotiert. Der Server *server.wan* ist mit einem langen Lichtwellenleiter direkt mit dem DSL Provider *provider.wan* verbunden. Alle anderen Verbindungen sind so kurz, dass idealisiert eine Länge von 0 m angenommen werden kann. Wegen Medienkonvertern in der DSL Verbindung zwischen Provider und Router ist die Ausbreitungsverzögerung dennoch verschieden von 0 s, was sich in einer gemessenen mittleren Round Trip Time von $RTT_{wp} = 10\,\text{ms}$ zwischen der Workstation und dem Provider wieder spiegelt. Gehen Sie davon aus, dass diese Zeit symmetrisch verteilt ist ($d_{Workstation \to Provider} = d_{Provider \to Workstation} = \frac{1}{2} RTT_{wp}$).Die IP-Adressen werden statisch zum Systemstart vergeben. Die Workstation *workstation.lan* kennt außerdem die IP des IP-Routers. Zusätzlich gelten die Konstanten gemäß Tabelle 1.

Ausbreitungsverzögerung	$c = 3 \cdot 10^8\ \frac{m}{s}$
Gemessene mittlere RTT	$RTT_{wp} = 10\,\text{ms}$
Paketlänge (MTU)	$L = 1250\ Byte$

Tabelle 1: Zusätzliche Konstanten

2 TCP + HTTP/1.1

Für die folgenden Leistungsanalyse wird nun angenommen, dass die Kommunikation mit dem Webserver *server.wan* auf HTTP/1.1 basiert. Des weiteren wird stets eine Seite mit zwei eingebetteten Objekten herunter geladen. Sowohl die Seite als auch die Objekte haben eine konstante Größe von L. Es treten keine Fehler auf. Gehen Sie idealisiert für Ihre Berechnungen davon aus, dass Pakete ohne Nutzdaten eine Größe von $0Byte$ besitzen. Es gilt weiterhin das in Abbildung 1 gezeigt Netzwerk. Verwenden Sie für Ihre Berechnungen die Konstanten aus Tabelle 1.

Berechnen Sie die Round Trip Time (RTT) zwischen der Workstation workstation.lan und dem Webserver. (Inklusive Formel!)
RTT = RTTwp + 2*dPropPS = 10ms + 2*$\frac{600km}{3*10^5 km/s}$ = 14ms

2.1.3

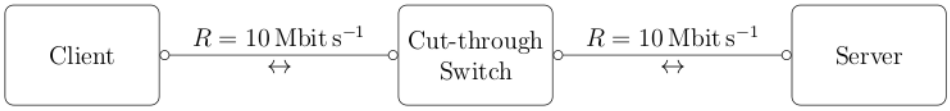


Abbildung 3: Netzwerktopologie

Sicherungsschicht Header	$h_2 = 25\,\mathrm{B}$
Netzwerkschicht Header	$h_3 = 50\,\mathrm{B}$
Transportschicht Header	$h_4 = 50\,\mathrm{B}$
Nutzdaten	$L = 1125\,\mathrm{B}$
Ausbreitungsverzögerung pro Link	$d_{prop} = 10\,\mu\mathrm{s}$

Tabelle 2: Zusätzliche Konstanten

2 Übertragungslatenzen (24 Punkte gesamt)

Gegeben sei die Netzwerktopologie gemäß Abbildung 3. Zwei Standard Ethernet (10Base-T) fähige Computer sind über einen sogenannten *Cut-through*-Switch miteinander verbunden. Im Unterschied zu einem klassischen *Store-and-Forward*-Switch leitet dieser die Pakete sobald wie möglich (= sobald der Empfänger bekannt ist) weiter, ohne den kompletten Empfang des Pakets abzuwarten.

Zusätzlich gelten die Konstanten gemäß Tabelle 2. Alle anderen relevanten Größen können zu 0 gesetzt werden. Gehen Sie vereinfachend davon aus, dass die einzelnen Header immer vollständig empfangen werden müssen. D.h. der Switch kann während er einen Header (h_2, h_3, h_4) empfängt keine Weiterleitungsentscheidung treffen.

Berechnen Sie die Round Trip Time (RTT) nur unter Berücksichtigung der in Tabelle 2 gelisteten Informationen für ein idealisiertes Paket der Gesamtgröße 0 B zwischen beiden Computern in Abbildung 3. Geben Sie ebenfalls eine Formel an.

Header = 25B + 50B + 50B = 125B = 1000b
$\frac{Header}{R} = \frac{1000b}{10*10^6 b/s}$ = 0,1ms
RTT = 2*(0,1ms + dprop + 0,1ms + dprop) = 2*(0.22ms) = 0.44 ms

2.1.4

2 TCP (26 Punkte gesamt)

Host A baut zu Host B eine TCP-Verbindung auf. Ohne eine konkrete Anfrage beginnt Host B so bald wie möglich mit dem Senden der Daten. Während des Sendens geht das komplette zweite Datenfenster verloren (siehe Abbildung 2). Als Überlastkontrolle wird nur Slow-Start und Timeout verwendet.

Host A beginnt das Senden mit der Sequenznummer $Seq = 10$ und Host B beginnt das Senden mit der Sequenznummer $Seq = 40$. Weiterhin gelten die Konstanten gemäß Tabelle 2.

Ausbreitungsverzögerung:	$d_{prop} = 45\,\mathrm{ms}$
Raten:	$R = 10\ \frac{Mbit}{s}$
Paketlänge:	$L = 1250\ Byte\ (= MSS)$
Timeout:	$\tau = 150\,\mathrm{ms}$

Tabelle 2: Konstanten

Die gemessene mittlere Round Trip Time (RTT) für die Verbindung zwischen Host A und Host B beträgt 2*dprop = 90 ms. Diese Messung wurde durchgeführt, als alle beteiligten Systeme ohne Last liefen (sofortiges Bearbeiten der Pakete). Da die Last bei Host B durch andere Dienste, die unabhängig vom Netzwerk sind, angestiegen ist, benötigt Host B nun im Mittel 10ms um Pakete zu bearbeiten. Andert sich dadurch etwas an der RTT?
Ja, die RTT wird größer.
RTT = 90ms + 10ms = 100ms

2.2 Datenaustausch

2.2.1

1.2 Datenaustausch (8 Punkte)

Gegeben sind zwei Hosts A und B, die über einen Link mit einer Bitrate $R = 1\,\mathrm{Gbit\,s^{-1}}$ und einer Länge von $l = 20\,000\,\mathrm{km}$ verbunden sind. Nehmen Sie an, dass die Ausbreitungsgeschwindigkeit $v = 2 \times 10^8\,\mathrm{m\,s^{-1}}$ beträgt.

Bestimmen Sie das Produkt b aus Bitrate und Ausbreitungsverzögerung. Was sagt dieser Wert aus?
$dprop = (2 * 10^7)/(2 * 10^8) = 0.1s$
$dprop * R = 0.1Gbit(Kanalpuffer)$

d = 0.1Gbit

2.3 Übertragungszeit

2.3.1

1.2 Datenaustausch (8 Punkte)

Gegeben sind zwei Hosts A und B, die über einen Link mit einer Bitrate $R = 1\,\mathrm{Gbit\,s^{-1}}$ und einer Länge von $l = 20\,000\,\mathrm{km}$ verbunden sind. Nehmen Sie an, dass die Ausbreitungsgeschwindigkeit $v = 2 \times 10^8\,\mathrm{m\,s^{-1}}$ beträgt.

Host A möchte nun ein Objekt der Größe O = 25 MB an Host B senden. Nehmen Sie an, dass die Nachricht ohne Unterbrechung (Wartezeit) versendet wird. Wie viele Bits können sich zu einem beliebigen Zeitpunkt maximal auf der Leitung befinden? Bestimmen Sie die gesamte Übertragungszeit. Nehmen Sie vereinfachend an, dass gilt: $1MB = 10^6B$.
Kanalpuffer = 0,1 Gbit = 0,0125GB = 12,5MB -> 200Mb können sich max. auf der Leitung befinden
dtrans = O/R = 0.2s
dprop = l/v = $\frac{20000km}{2*10^8m/s}$= 0.1s
dges = 0.2s + 0.1s = 0.3s

2.3.2

Bei einem Experiment sollen Daten im Meer via Schall übertragen werden. Knoten A und B sind 150m von einander entfernt. Die Ausbreitungsgeschwindigkeit auf dem Link beträgt 1,5 km/s. A sendet B 1,0 kByte mit einer Datenrate von 80 kbit/s.
Es treten keine weiteren Verzögerungen auf. Berechnen Sie eine erste Abschätzung der Übertragungszeit d für diese Annahmen.
d = dProp + dTrans = l/v + L/R = 1/10 s + 1/10 s = 2/10 s

2.3.3

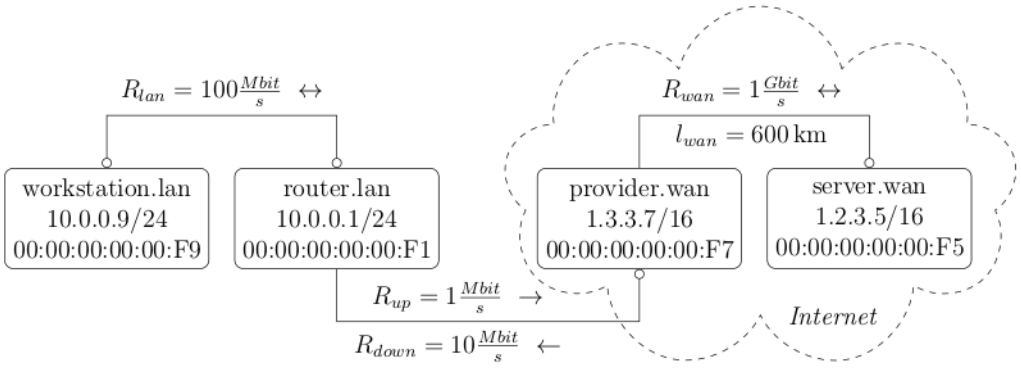


Abbildung 1: Netzwerkkonfiguration

Szenario

Das hier gezeigte Szenario, sowie die hier gelisteten Konstanten gelten für alle folgenden Aufgaben.

Gegeben sei eine Netzwerkkonfiguration gemäß Abbildung 1. Sie besteht aus einem Local Area Network (LAN) und aus einem Wide Area Network (WAN). Beide Netzwerke sind mittels einer asymmetrischen Digital Subscriber Line (DSL) verbunden. Der Router *router.lan* betreibt Paket Weiterleitung und eine Domain Name System (DNS). Jedes Rechteck bildet einen Teilnehmer ab. In der ersten Zeile steht der Domain Name des Teilnehmers. In der zweiten Zeile seine IP und in der dritten Zeile seine Medium Access Control (MAC) Adresse. Die WAN-Informationen des Router Interfaces *router.lan* sowie die Informationen für das zweite Interface von der Gegenstelle *provider.wan* sind nicht gelistet, da sie für diese Aufgabe keine Relevanz haben. (Die gezeigten Informationen gelten für den Link mit Punkt.) An den Links sind jeweils die Verbindungsgeschwindigkeiten in die jeweilige Richtung (Pfeile) annotiert. Der Server *server.wan* ist mit einem langen Lichtwellenleiter direkt mit dem DSL Provider *provider.wan* verbunden. Alle anderen Verbindungen sind so kurz, dass idealisiert eine Länge von 0 m angenommen werden kann. Wegen Medienkonvertern in der DSL Verbindung zwischen Provider und Router ist die Ausbreitungsverzögerung dennoch verschieden von 0 s, was sich in einer gemessenen mittleren Round Trip Time von $RTT_{wp} = 10\,\text{ms}$ zwischen der Workstation und dem Provider wieder spiegelt. Gehen Sie davon aus, dass diese Zeit symmetrisch verteilt ist ($d_{Workstation \to Provider} = d_{Provider \to Workstation} = \frac{1}{2} RTT_{wp}$).Die IP-Adressen werden statisch zum Systemstart vergeben. Die Workstation *workstation.lan* kennt außerdem die IP des IP-Routers. Zusätzlich gelten die Konstanten gemäß Tabelle 1.

Ausbreitungsverzögerung	$c = 3 \cdot 10^8\ \frac{m}{s}$
Gemessene mittlere RTT	$RTT_{wp} = 10\,\text{ms}$
Paketlänge (MTU)	$L = 1250\ Byte$

Tabelle 1: Zusätzliche Konstanten

2 TCP + HTTP/1.1

Für die folgenden Leistungsanalyse wird nun angenommen, dass die Kommunikation mit dem Webserver *server.wan* auf HTTP/1.1 basiert. Des weiteren wird stets eine Seite mit zwei eingebetteten Objekten herunter geladen. Sowohl die Seite als auch die Objekte haben eine konstante Größe von L. Es treten keine Fehler auf. Gehen Sie idealisiert für Ihre Berechnungen davon aus, dass Pakete ohne Nutzdaten eine Größe von $0Byte$ besitzen. Es gilt weiterhin das in Abbildung 1 gezeigt Netzwerk. Verwenden Sie für Ihre Berechnungen die Konstanten aus Tabelle 1.

Wie lange dauert es ein Paket der Größe L von der Workstation zum Webserver (dup) und vom Webserver zur Workstation (ddown) zu übertragen? Geben Sie jeweils Formel und Zahlenwert an. Tragen Sie einen der beiden Abläufe in Abbildung 2 ein. Berücksichtigen Sie qualitativ die zeitliche Ausdehnung der Pakete auf den verschiedenen Links.

dup = 0/c + 0/c + 600km/c + L/100$\frac{Mbit}{s}$ + L/1* $\frac{Mbit}{s}$ +L/1* $\frac{Gbit}{s}$ = 2ms + 0.1ms + 10ms + 0.01ms = 12.11ms

2.3.4 1 Paket

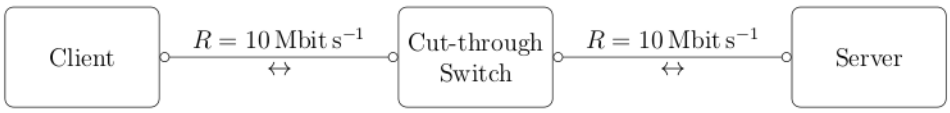


Abbildung 3: Netzwerktopologie

Sicherungsschicht Header	$h_2 = 25\,\mathrm{B}$
Netzwerkschicht Header	$h_3 = 50\,\mathrm{B}$
Transportschicht Header	$h_4 = 50\,\mathrm{B}$
Nutzdaten	$L = 1125\,\mathrm{B}$
Ausbreitungsverzögerung pro Link	$d_{prop} = 10\,\mu\mathrm{s}$

Tabelle 2: Zusätzliche Konstanten

2 Übertragungslatenzen (24 Punkte gesamt)

Gegeben sei die Netzwerktopologie gemäß Abbildung 3. Zwei Standard Ethernet (10Base-T) fähige Computer sind über einen sogenannten *Cut-through*-Switch miteinander verbunden. Im Unterschied zu einem klassischen *Store-and-Forward*-Switch leitet dieser die Pakete sobald wie möglich (= sobald der Empfänger bekannt ist) weiter, ohne den kompletten Empfang des Pakets abzuwarten.

Zusätzlich gelten die Konstanten gemäß Tabelle 2. Alle anderen relevanten Größen können zu 0 gesetzt werden. Gehen Sie vereinfachend davon aus, dass die einzelnen Header immer vollständig empfangen werden müssen. D.h. der Switch kann während er einen Header (h_2, h_3, h_4) empfängt keine Weiterleitungsentscheidung treffen.

Wie lange dauert es L Nutzdaten (= ein Paket) zwischen den Computern zu übertragen (einfacher Weg)? Geben Sie jeweils eine allgemeine Formel für das Cut-through(dct) und für das Store-and-Forward(dsf) Verfahren unter Berücksichtigung einer variablen Anzahl von Links E an. Um wie viel Prozent ist hier (Abbildung 3,E = 2) welches Verfahren schneller?

dsf = E*dprop + E*dtrans = E*l/v + E*$\frac{h+L}{R}$ = 2*0.01ms + 2*$\frac{10000b}{10*10^6 b/s}$= 0.02ms + 2ms = 2.02ms

dcs = E*dprop + dtrans+ (E-1)*dtransHeader = 2*0.01ms + 1ms + 1*$\frac{1000b}{10*10^6 b/s}$ = 1.01ms + 0.1ms = 1.02 ms

Antwort = 2.02/1.02 = 1.98 => das Verfahren ist 98% schneller

2.3.5 O/L Pakete

2 Latenzzeiten (21 Punkte)

Abbildung 2: Beispiel Netzwerk mit $E = 3$ Links

Abbildung 2 zeigt eine beispielhafte Netzwerktopologie, in dem sich zwischen Host A und Host B 2 weitere Knoten und somit $E = 3$ Links befinden. Die Rate jedes Links beträgt R. Im folgenden ist die Anzahl der Links E zwischen den Hosts variabel. Ein Objekt der Größe O soll von Host A auf Host B übertragen werden. Pro Paket werden L Byte Nutzdaten und zusätzlich h Byte Header übertragen.

Tipp: Store-and-Forward berücksichtigen!

Wie muss die Größe der Nutzdaten L in Relation zu der Objektgröße O gewählt werden, damit die Übertragungszeit dmin möglichst klein ist? Nehmen Sie vereinfachend an, dass immer gilt: $O/L \in N$

d = E*dprop+O/L*dtrans+(E-1)*dtrans = E*dprop+O/L*$\frac{h+L}{R}$+(E-1)*$\frac{h+L}{R}$= E*dprop + Oh/LR+O/R+(E-1)*(L/R)+(E-1)*h/R

=> Alles ohne L ist irrelevant : Oh/LR+L/R+(E-1)

=> Ableiten: $-Oh/L^2R + (E-1)/R$

=> gleich 0 setzen: $\frac{E-1}{R} = \frac{Oh}{L^2*R}$

$L^2 = (\frac{Oh}{E-1})$

$L = \sqrt{Oh/(E-1)}$

Wie muss die Größe der Nutzdaten L in Relation zu der Objektgröße O gewählt werden,damit die Übertragungszeit dmax möglichst groß ist? (Aber: dmax != unendlich). Vernachlässigen Sie für diese Aufgabe die hByte Header (h = 0). Geben Sie eine allgemeine Formel für die Übertragungszeit dmax in Abhängigkeit von E und ohne L an.

d = E*dprop+O/L*dtrans+(E-1)*dtrans= E*dprop + O/R + (E-1)(L/R)

=> L = O (da h = 0) : E*dprop + O/R + (E-1)(O/R) = E*dprop + E*(O/R)

2.3.6 O/L Pakete und GO-Back-N

1.3 Latenzzeit (4 + 6 + 3 Punkte)

Ein Objekt der Größe $O = 1\,\text{MB}$ soll über ein Netzwerk mit 5 dazwischen liegenden *Store-and-Forward*-Routern ($E = 6$ Links) übertragen werden. Die Paketgröße beträgt $L = 1\,\text{kB}$. Die Übertragungsrate jedes Links beträgt $R = 8\,\text{Mbit}\,\text{s}^{-1}$. (Vereinfachung: $1\,\text{MB} = 10^3\,\text{kB} = 10^6\,\text{B}$. Es gibt keine Fehler und auch keine Ausbreitungsverzögerung)

Wie lange dauert es (d1) das Objekt O zu übertragen, wenn keine Übertragungsprotokoll zugrunde liegt (keine ACKs)?
d1 = E*dprop + O/L*dtrans + (E-1)*dtrans = 0+ 1000* $\frac{1*10^3*8b}{8*10^6 b/s}$+ 4*$\frac{1*10^3*8b}{8*10^6 b/s}$ = 1s+0.004s = 1.004s
Gehen Sie jetzt davon aus, dass Go-Back-N mit einer festen Fernstergröße von W = 5 für die Übertragung verwendet wird. ACKs haben eine idealisierte Größe von 0. Wie lange dauert es (d2) bis das Objekt O vollständig auf dem Zielhost verfügbar ist? Wie viele Wartezeiten P werden benötigt und wie lange dauert eine Wartezeit dp. (Hinweis: Es gelten weiterhin die unter Abschnitt 1.3 genannten Größen. E = 6 Links, ...)
Allgemeine Formel für mehrere Links : $\frac{E*L/R}{L/R}$
Fenster groß genug?: $\frac{6*1ms}{1ms} = 6$
=> 5<6
=> S = $\frac{W}{\frac{E*L/R}{L/R}} = 5/6 = 0.8$ Antwort: O/R * 0.8 = 0.8s
P= ??????????????????????????????

2.3.7 Durchsatz GO-Back-N und Stop and Wait

2.2 Durchsatz

Bestimmen Sie den normierten Durchsatz für ein Netzwerk mit folgenden Parametern: Ausbreitungsverzögerung D = 5 ms, Paketgröße L = 1000 Bits, Datenrate R = 1 Mbps, Fehlerwahrscheinlichkeit p = 0.03 und Fenstergröße W = 4 Pakete.

Für Stop-And-Wait: Allgemeine Formel : S = $\frac{1-p}{1+2a}$
S = $\frac{1-p}{1+2RD/L} = \frac{1-0.03}{1+10s}$= 8.8%
Fur Go-Back-N:
Fenster groß genug?: 1+2a
1+2a = 11
11>4
Allgemeine Formel(falls W<1+2a) = $\frac{W(1-p)}{(1+2a)*(1-p+Wp)}$
S = $\frac{0{,}97*4}{(0{,}97+4*0{,}03)(1+0{,}97*10)}$= 32%

2.3.8

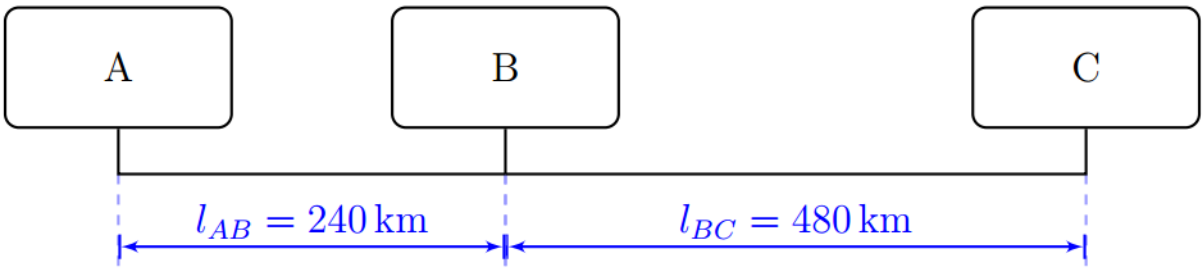


Abbildung 1: Netzwerkkonfiguration

Folgendes gilt: Drei Hosts A, B und C seien uber einen Netzwerk-Bus gemäß Abbildung 1 miteinander verbunden. Die Bitrate beträgt R = 125 kbit/s. Die Ausbreitungsgeschwindigkeit beträgt v = 2 * 108 m/s . Die Abstände entnehmen Sie Abbildung 1. Bestimmen Sie die Ausbreitungsverzögerungen dAB, dBC und dAC zwischen den einzelnen Hosts in Bitzeiten. (Hinweis: Berechnen Sie zunächst die Ausbreitungsverzögerung in der üblichen Einheit Sekunden. Wie aus der Übung bekannt entspricht eine Bitzeit der Zeit, die benötigt wird um ein Bit auf die Leitung zu legen. Bei der hier gegebenen Bitrate R gilt also: 1 Bitzeit = 0.008 ms)
Allgemeine Formel: dprop = l/v
dAB = $240km/2 * 10^8 m/s = 1,2ms$
dBC = 2,4ms
dAC = 3,6ms
Bitzeiten: $1,2/8 * 10^{-3} = 150$
300
450

2.3.9 1 Paket und O/L Pakete

1.3 Latenzzeit (2 + 2 + 3 + 3 Punkte)

Host A möchte ein Objekt O an Host B senden. Zwischen Host A und Host B liegt ein weiterer *Store-and-Forward*-Host M. Eine 30 km lange Glasfaserleitung verbindet jeweils Host A mit Host M und Host M mit Host B (Netztopologie: $A \leftrightarrow M \leftrightarrow B$). Informationen breiten sich auf den Leitungen mit einer Geschwindigkeit von $v = 3 \times 10^8\,\text{m}\,\text{s}^{-1}$ aus. Die Paketgröße beträgt $L = 1\,\text{kB}$. Die Übertragungsrate jedes Links beträgt $R = 8\,\text{Mbit}\,\text{s}^{-1}$. (Vereinfachungen: $1\,\text{kB} = 10^3\,\text{B}$. Es gibt keine Fehler und es weden keine Pakete verworfen.)

Bestimmen Sie die Ausbreitungsverzögerung (dprop) eines Links.
Allgemeine Formel: dprop = l/v
dprop =$30km/(3 * 10^8 ms) = 0.1ms$
Bestimmen Sie die Übertragungsverzögerung (dtrans) eines Pakets auf einem Link. Allgemeine Formel: dtrans = L/R
'dtrans =$\frac{1*10^3*8b}{8*10^6 b/s}$ =1ms
Wie lange dauert es (d1) ein Objekt der Größe O = 1 kB von A nach B zu ubertragen?
L = O => es gibt nur 1 Paket
Allgemeine Formel : E*(dprop*dtrans)
d1 = 2*(0.1ms+1ms) = 2.2ms
Wie lange dauert es (d2) ein Objekt der Größe O = 3.5 kB von A nach B zuübertragen?
N = O/L = 3.5(Pakete)
Allgemeine Formel : E*dprop + O/L*dtrans + (E-1)*dtrans
d2 = 0.2ms + 3.5ms+1ms = 4.7ms

2.4 Cyclic Redundancy Check

2.3 Cyclic Redundancy Check

Gegeben ist eine Nutzdatengröße von 8 Bit, sowie folgende Generatorpolynom:
$G(x) = x^4 + x^2 + x + 1 = (x + 1)(x^3 + x^2 + 1)$.

Berechnen Sie die Prufdaten für folgende Nutzdaten: 11100011
Allgemeines Vorgehen : Stellen werden verXORd => Stellen werden an die Nutzdaten angehaengt
111000110000 : 10111
10111
10110110000
10111
1110000
10111
101100
10111
10

Ergebnis: 111000110010
Nehmen Sie an, dass ein Sender folgendes Bitmuster versendet: 100001110101. Ein Empfänger empfängt diese Daten als: 100010001110. Kann die Störung auf dem Kanal zuverlässig erkannt werden? Begrunden Sie ihre Antwort. Allgemeine Formel : (D,R)/G != 0
Hier schnelle Begründung: 5 Zusammenhängende Fehler. Da die Anzahl an Fehlern un-

gerade ist wird der Fehler auf jeden Fall erkannt

2.5 HTTP Antwortzeiten

2.2 Durchsatz

Bestimmen Sie den normierten Durchsatz für ein Netzwerk mit folgenden Parametern: Ausbreitungsverzögerung D = 5 ms, Paketgröße L = 1000 Bits, Datenrate R = 1 Mbps, Fehlerwahrscheinlichkeit p = 0.03 und Fenstergröße W = 4 Pakete.

Nicht-persistentes HTTP
M+1 = Anzahl der Bilder + Basisseite
Allgemeine Formel: (M+1)*2RTT = 82 RTT
Persistentes HTTP mit Pipelining
3 RTT (2 für Basis, 1 für Bilder)
Nicht-persistentes HTTP mit 10 parallelen Verbindungen
Allgemeine Formel: (M/X+1)*2RTT
=> (4+1)*2RTT =10

2.6 Kollisionserkennung CSMA/CD

Rahmensendedauer ($L = 1000\,\text{bit}$)	$d_{trans} = 1000$ Bitzeiten
Ausbreitungsverzögerung	$d_{AB} = 150$ Bitzeiten $d_{BC} = 300$ Bitzeiten $d_{AC} = 450$ Bitzeiten
Dauer, die der Kanal vor dem tatsächlichen Senden frei sein muss	$t_{sense} = 50$ Bitzeiten
Jam-Signal-Sendedauer	$t_{jam} = 50$ Bitzeiten
Backoff-Konstanten	$K_A = 3$ $K_B = 2$ $K_C = 1$
Backoff-Zeitslot	$\tau = 700$ Bitzeiten

Tabelle 1: Netzwerktopologie

Stellt die gewählte Rahmengröße von L = 1000 bit in dem hier gezeigten Netzwerk sicher, dass Kollisionen zuverlässig detektiert werden können? Wie groß darf die maximale Ausbreitungsverzögerung (in Bitzeiten) für die hier gegebenen Parameter zwischen 2 Hosts sein. Geben Sie auch den formalen Zusammenhang an.
Allgemeine Formel für sicheres Erkennen: L/R>2*D
$\frac{1000b}{125*10^3b} > 2 * 450 * 8 * 10^{-3}ms$
8ms > 7,2ms => Rahmen ist groß genug

2.7 Signalabtastung

Wir wollen ein analoges Signal durch periodisches Abtasten wieder vollständig durch die Abtastwerte zurückgewinnen. Welche Frequenz darf somit ein analoges Signal maximal haben, wenn es alle 0.2 ms abgetastet wird?
fa = 1/ (0,2ms) = 0.4ms = 2.5kHz

2.8 Bandbreite

Bei einer Datenübertragung soll eine maximale Datenrate von 96 kbit/s.Wie groß muss die Bandbreite B gewählt werden, wenn man Rauschen vernachlässigt und wenn pro Schritt ein Signal aus vier Bits übertragen werden kann? Wie verändert sich die Bandbreite B, wenn wir ein Signal-Rausch-Verhältnis von S/N = 15 annehmen?
Nach Shannon:
B = mD/(log2 (1+ S/N)) = $\frac{96kb/s}{4}$= 24 kb/s

2.9 Brutto Übertragungsrate

2.9.1

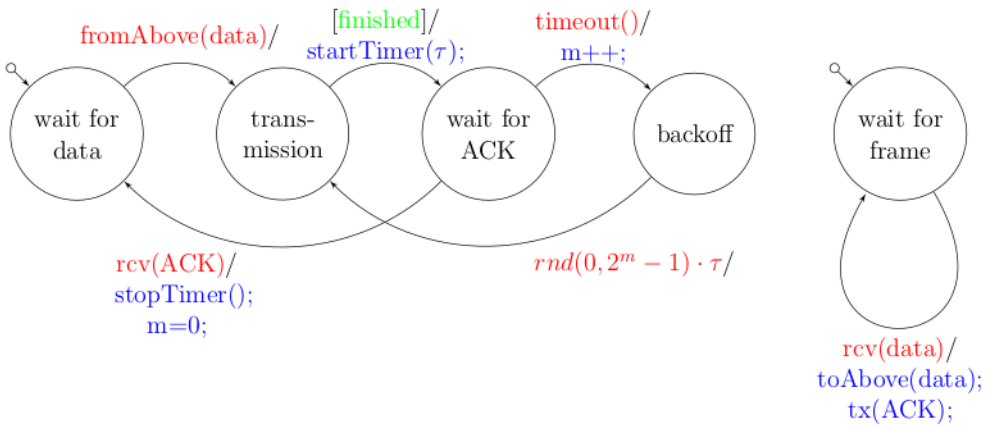


Abbildung 3: *ALOHA* - Zustandsdiagramm

Im folgenden befinden sich nun zwei weitere Teilnehmer im LAN (insgesamt 4 Stationen). Deshalb wird die drahtgebundene Verbindung durch eine kabellose ersetzt. Sie basiert auf dem ALOHA Medienzugriffsverfahren und weist wie vorher eine effektive Rate von Rlan = 100Mbit/s auf. Abbildung 3 zeigt das Verhalten der einzelnen Teilnehmer. Nehmen Sie an, dass diese im Mittel mit einer Wahrscheinlichkeit von Ptx= 0.2 senden.
Wie hoch muss die Übertragungsrate der physikalischen Schicht Rtx mindestens sein, damit die erwünschte effektive Rate Rlan= 100 Mbit/s eingehalten werden kann? Begründen sie ihre Antwort mit einer Formel.
S = Durchsatz
N = 4
p = 0.2
Allgemeine Formel: $S = Np(1-p)^{2(N-1)}$
$\Rightarrow S = 4 * 0.2(1-0.2)^{2*(4-1)} = 0.2$
$\Rightarrow R * 102 = 100Mb/s * 1,2 = 120Mb/s$

2.10 Leistungsanalyse

2.10.1 Überlastfenstervergrößerung

2.4 TCP-Leistungsanalyse

Ein Client baut zu einem Server eine TCP-Verbindung auf und lädt ein Objekt O herunter. Die Paketgröße (MTU) beträgt L, die Datenrate ist R und die Round-Trip-Time ist RTT.

Nehmen Sie an, TCP vergrößert sein Überlastfenster bei jedem empfangenem ACK-Segment um 3 anstelle von ursprünglich 1 MTU. Somit besteht das erste Fenster aus einem Segment, das zweite aus 4 Segmenten, das dritte aus 16 Segmenten, usw.

Tipp: $\sum_{i=1}^{N} 4^{(i-1)} = \frac{4^N-1}{3}$

Bestimmen Sie K (Anzahl der Fenster) als Ausdruck von O und L.
K = min[k:$\frac{4^k-1}{3}$ >= O/L] = min[k:k >=log4(3*O/L+1)] = aufrunden(log4(3*0/L+1))
Bestimmen Sie Q (Anzahl der Slow-Start-Wartezeiten) als Ausdruck von RTT, L und R.
Q = $max[k : 4^{k-j} * L/R <= L/R + RTT] = max[k : k <= log4(1 + \frac{RTT}{L/R}) + 1] = abrunden(log4(1 + \frac{RTT}{L/R})) + 1$

2.10.2 easy

2.2 TCP Signalisierung (10 Punkte)

In Abbildung 2 baut ein Host A zu einem Host B eine TCP Verbindung auf. Nach einer Anfrage mit einer Größe von 5 B beginnt Host B mit maximaler Geschwindigkeit Daten an Host A zu übertragen. Vervollständigen Sie das in Abbildung 2 gezeigte Ablaufdiagramm durch Eintragen der Sequenznummern (Seq), der Acknowledgment Nummern (Ack) und des Congestion Windows (CW). Gehen Sie davon aus, dass die Puffer immer unendlich Groß sind. Die Fenstergrößen sind nach Erhalten und vor dem Aussenden eines Segments in der Maximum Segment Size (MSS) Einheit anzugeben, wobei gilt: 1 MSS = 10 B. Host A startet mit Seq=50 und Host B mit Seq=0. Es gibt kein Selective Acknowledgment (SACK) und keinen weiteren Timeout. Zu welcher Zeit (t_1 oder t_2) könnte Host B neue Daten übermitteln? Begründen Sie Ihre Antwort.

Bis zu welchem Fenster Q treten bei einem unendlich großen Objekt O Wartezeiten auf?
Allgemeine Formel:)(Abrunden(log2(1+ [RTT/ (L/R)])))+1
$\frac{RTT}{L/R} = \frac{2s}{80ms}$= 25s
Q = abrunden(4.5)+1 = 5

Wie viele Fenster K werden benötigt, um das Objekt O1 zu ubertragen?
Allgemeine Formel: aufrunden(log2(O/L +1))
$\frac{O}{L} = \frac{1*10^6}{1*10^3} = 1000$
K = aufrunden(9.9)= 10
Wie viele Wartezeiten P treten bei der Übertragung des Objekts O1 auf?
Allgemeine Formeln = min(Q,K-1)
P = min(5,9) = 5

2.10.3 Durch Verlauf

2 TCP (26 Punkte gesamt)

Host A baut zu Host B eine TCP-Verbindung auf. Ohne eine konkrete Anfrage beginnt Host B so bald wie möglich mit dem Senden der Daten. Während des Sendens geht das komplette zweite Datenfenster verloren (siehe Abbildung 2). Als Überlastkontrolle wird nur Slow-Start und Timeout verwendet.

Host A beginnt das Senden mit der Sequenznummer $Seq = 10$ und Host B beginnt das Senden mit der Sequenznummer $Seq = 40$. Weiterhin gelten die Konstanten gemäß Tabelle 2.

Ausbreitungsverzögerung:	$d_{prop} = 45\,\text{ms}$
Raten:	$R = 10\ \frac{Mbit}{s}$
Paketlänge:	$L = 1250\ Byte\ (= MSS)$
Timeout:	$\tau = 150\,\text{ms}$

Tabelle 2: Konstanten

Im folgenden verwendet TCP ausschließlich Stop-and-Wait bei der Übertragung von Daten. Der Verbindungsaufbau erfolgt weiterhin nach dem bekannten 3-Wege-Handshake.Berechnen Sie auf dieser Grundlage die Latenzzeit, die benötigt wird, um ein Objekt O = 5000 Bytes im fehlerfreien Fall zu ubertragen (Zeit bis die Daten bei Host A verfügbar sind). Die benötigten-Konstanten entnehmen Sie bitte Tabelle 2 sowie der Aufgabe 2.1. (RTT = 100ms)
Allgemeine Formel: 2*RTT(Verbindungsaufbau)+O/R(Übertragung des Objektes)
siehe Übung 5 S.4-6
1 RTT (100ms) (Pfeile 1 und 2 in der Zeichnung)
+ 55 ms (Pfeil 3 der Zeichnung,dprop + 10ms(Wartezeit)),)
+ 3 * (100ms +1ms) (die ersten 3 Pakete, 1ms für die Übertragung wie 100 ms für die RTT)
+ 56 ms (das letzte Paket ohne den ACK-Pfeil zurück , da man wegen der Aufgaben-

stellung das letzte ACK nicht braucht)
= 504 ms

2.10.4

3.3 Leistungsanalyse [13 Punkte]

Im Folgenden soll die Verzögerungszeit zwischen Verbindungsaufbau und vollständigem Eintreffen der ersten E-Mail der Größe O bei Host C ermittelt werden. Wie bereits in Aufgabe 3.2 angenommen, können Sie davon ausgehen, dass nur Pakete, die die eigentliche E-Mail enthalten, eine Größe verschieden von 0 aufweisen.

Modifizieren Sie die aus der Vorlesung bekannte Formel für die TCP-Latenzberechnung mit Slow-Start (1) so, dass die POP3-Anwendung (wie in Aufgabe 3 spezifiziert) berücksichtigt wird. Berücksichtigen Sie auch die für die Berechnung benötigten Wartezeiten $P = min(Q, K-1)$, wobei Q die Anzahl der Wartezeiten für ein unendlich großes Objekt und K die Anzahl der Sendefenster darstellt. Vereinfachen Sie die Formeln für Q und K (geometrische Reihe: $\sum_{k=0}^{m} a_0 q^k = a_0 \frac{q^{m+1}-1}{q-1}$). Welche Rate aus Tabelle 1 muss für die Formel 1 verwendet werden?

Berechnen Sie damit die Latenzzeit, die benötigt wird, um ein E-Mail der Größe $O = 9000\,\text{Byte}$ zu übertragen (Zeit bis die Daten verfügbar sind). Die benötigten Konstanten entnehmen Sie bitte Tabelle 1 sowie der Aufgabe 3.1.

Tipp: Das erste Sendefenster der E-Mail hat eine Größe verschieden von 1 MSS.

$$d = 2RTT + \frac{O}{R} + P\left(RTT + \frac{L}{R}\right) - \left(2^P - 1\right)\frac{L}{R} \tag{1}$$

2.10.5

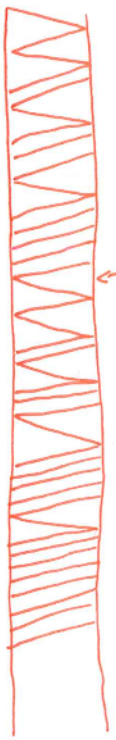
b) (8 Punkte) Eine Webseite soll mit nicht-persistentem HTTP übertragen werden. Die Webseite besteht aus einer HTML-Basisseite der Größe **7 S** mit einem einzigen eingebetteten Objekt der Größe **14 S**, wobei **S** die Größe eines TCP-Segments in Bits ist. Die dynamische Fenstergröße (in Anzahl von Segmenten) einer TCP-Verbindung soll sich ausschließlich nach dem Slow-Start-Mechanismus entwickeln. Die Übertragungsrate **R** und die Round Trip Time **RTT** werden als idealisiert konstant angenommen.

- Berechnen Sie die Antwortzeit der Übertragung (in Abhängigkeit von **S** und **R**) für den Fall, dass **RTT = 4 S/R** gilt.
- Veranschaulichen Sie das Übertragungsverhalten an Hand einer Skizze, aus der hervorgeht, wann Wartezeiten auftreten und in welchen Fenstern wieviele Segmente übertragen werden!
- In welchem Verhältnis stehen die Latenzen für das eingebettete Objekt und die halb so große HTML-Basisseite?

allg. Delay-Formel: $D = 2RTT + \frac{O}{R} + P\left(RTT + \frac{S}{R}\right) - \left(2^P - 1\right)\frac{S}{R}$

mit $P = \min(Q, K-1)$ und

$Q = \lfloor ld\left(1 + \frac{RTT}{S/R}\right) \rfloor + 1$ $\quad K = \lceil ld\left(\frac{O}{S} + 1\right) \rceil$



- HTML-Basisseite: $O = 7S$ $\quad RTT = 4\,S/R$

$Q = \lfloor ld(5) \rfloor + 1 = 3$, $\quad K = \lceil ld(8) \rceil = 3 \Rightarrow P = K - 1 = 2$

$D_{Basis} = 2RTT + 7\,S/R + 2RTT + 2\,S/R - 3\,S/R = 4RTT + 6\,S/R = 22\,S/R$

- eingebettetes Objekt: $O = 14S$ $\quad RTT = 4\,S/R$

$Q = \lfloor \log_2(5) \rfloor + 1 = 3$ $\quad K = \lceil ld(15) \rceil = 4 \Rightarrow P = Q = 3$

$D_{eino} = 2RTT + 14\,S/R + 3RTT + 3\,S/R - 7\,S/R =$

$= 5RTT + 10\,S/R = 30\,S/R$

$\frac{D_{eino}}{D_{Basis}} = \frac{30\,S/R}{22\,S/R} \approx 1{,}4$

Hinweis: wir verwenden jetzt L statt S

A: Das Verhältnis ist ungefähr 1,4.

Seite 7

2.10.6

bei uns ist $S \hat{=} L$

c) (9 Punkte) Eine Objekt der Größe **O** (in bits), bestehend aus einer geraden Zahl **O/S** von Segmenten, soll über eine TCP-Verbindung kopiert werden. Unter der Annahme einer dynamischen Fenstergröße, die sich nach dem Slow-Start-Mechanismus entwickelt (wie in der Vorlesung), wird zunächst die erste Hälfte der Segmente erfolgreich übertragen. Die nachfolgenden Segmente gehen verloren, bis der Fehler durch die TCP-Fehlerkontrolle erkannt wird. Danach finden keine Datenverluste mehr statt. Die Übertragungsrate **R** und die Round Trip Time **RTT** werden als idealisiert konstant angenommen.

Acks OK

↳ v. ∞ großem Objekt ausgehen

Wie viele Slow-Start-Wartezeiten **P** entstehen bei der Übertragung des gesamten Objekts?
(Es darf angenommen werden, dass schon die Hälfte des Objekts hinreichend groß ist.)

Berechnen Sie die Latenz für das Kopieren des gesamten Objekts in Abhängigkeit von den gegebenen Größen.
(Veranschaulichen Sie sich das Übertragungsverhalten an Hand einer groben Skizze!)

$$Q_1 = \lfloor \mathrm{ld}\left(1 + \frac{RTT}{S/R}\right) \rfloor + 1$$

$$P = 2Q_1$$

- Übertragungslatenz 1. Hälfte

$$D_1 = 2RTT + \frac{O}{2R} + Q_1\left[RTT + \frac{S}{R}\right] - \left(2^{Q_1} - 1\right)\frac{S}{R}$$

- 2. Hälfte

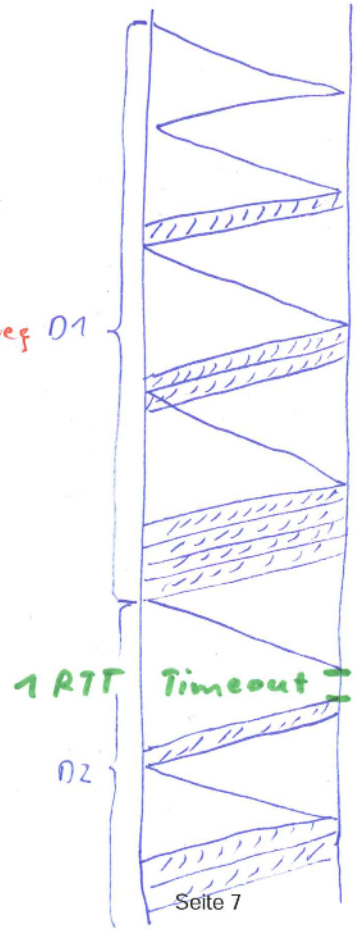
1 RTT ≠ wg. Handshake weg D1

$$D_2 = \text{Timeout} + RTT + \frac{O}{2R} + Q_1\left[RTT + \frac{S}{R}\right] - \left(2^{Q_1} - 1\right)\frac{S}{R}$$

RTT (idealisiert)

$$= 2RTT + \frac{O}{2R} + Q_1\left[RTT + \frac{S}{R}\right] - \left(2^{Q_1} - 1\right)\frac{S}{R}$$

$$D = D_1 + D_2 = 4RTT + \frac{O}{R} + P\left[RTT + \frac{S}{R}\right] - 2\frac{S}{R}\left(2^{Q_1} - 1\right)$$



2.11 Harmonische Schwingunge

1.7 Harmonische Schwingungen (5 Punkte)

In einem neuartigen Telefonnetz besteht ein periodisches Signal aus 2 B. Bestimmen Sie für die Bitraten $B_1 = 2400\,\mathrm{bit\,s^{-1}}$, $B_2 = 4800\,\mathrm{bit\,s^{-1}}$ und $B_3 = 9600\,\mathrm{bit\,s^{-1}}$ die Frequenz der ersten harmonischen Schwingung. Berechnen Sie die Bandbreitenbeschränkung f_c, wenn bei der Bitrate B_1 insgesamt 20 harmonische Schwingungen übertragen wurden.

Allgemein: f = 1/T
fi = Bi/(2*8)
fi1 = 150hz
fi2 = 300hz
fi3 = 600hz
Allgemein: fc/f = Anzahl harmonischer Schwingungen
fc = Anzahl*fi1 = 20*150hz = 300hz

2.12 Übertragungsvergleich

1.6 Marathonläufer (5 Punkte)

Ein Marathonläufer trägt eine Schachtel von 5 CD-ROMs mit einer Speicherkapazität von jeweils 700 MB mit sich. Bei einem Marathon erreicht der Läufer eine Durchschnittsgeschwindigkeit von $21.6\,\mathrm{km\,h^{-1}}$. Bis zu welcher Entfernung überträgt der Marathonläufer die Daten schneller als eine DSL-Verbindung, wenn diese eine Datenrate von $4\,\mathrm{Mbit\,s^{-1}}$ beziehungsweise $16\,\mathrm{Mbit\,s^{-1}}$ hat? Gehen Sie von einer unendlich schnellen Ausbreitungsverzögerung der DSL Leitung aus. Das brennen der CD-ROMs nimmt keine Zeit in Anspruch. Nehmen Sie vereinfachend für Ihre Rechnung an, dass gilt: $1\,\mathrm{MB} = 10^6\,\mathrm{B}$.

Lcd $= 5 * 700MB = 3500 * 10^6 B$
$Rmara = 21.6 * 10^3 m/h$
$Rdsl1 = 16 * 10^6 b/s$
$Rdsl2 = 4 * 10^6 b/s$
ddsl1 $= \frac{3500MB}{2MB/s} = 1750$s
ldsl1 $= 1750s * 21.6km/h = 1750s * 6m/s = 10.5km$
ddsl1 $= \frac{3500MB}{0.5MB/s} = 7000$s
ldsl1 $= 7000s * 21.6km/h = 7000s * 6m/s = 42.5km$

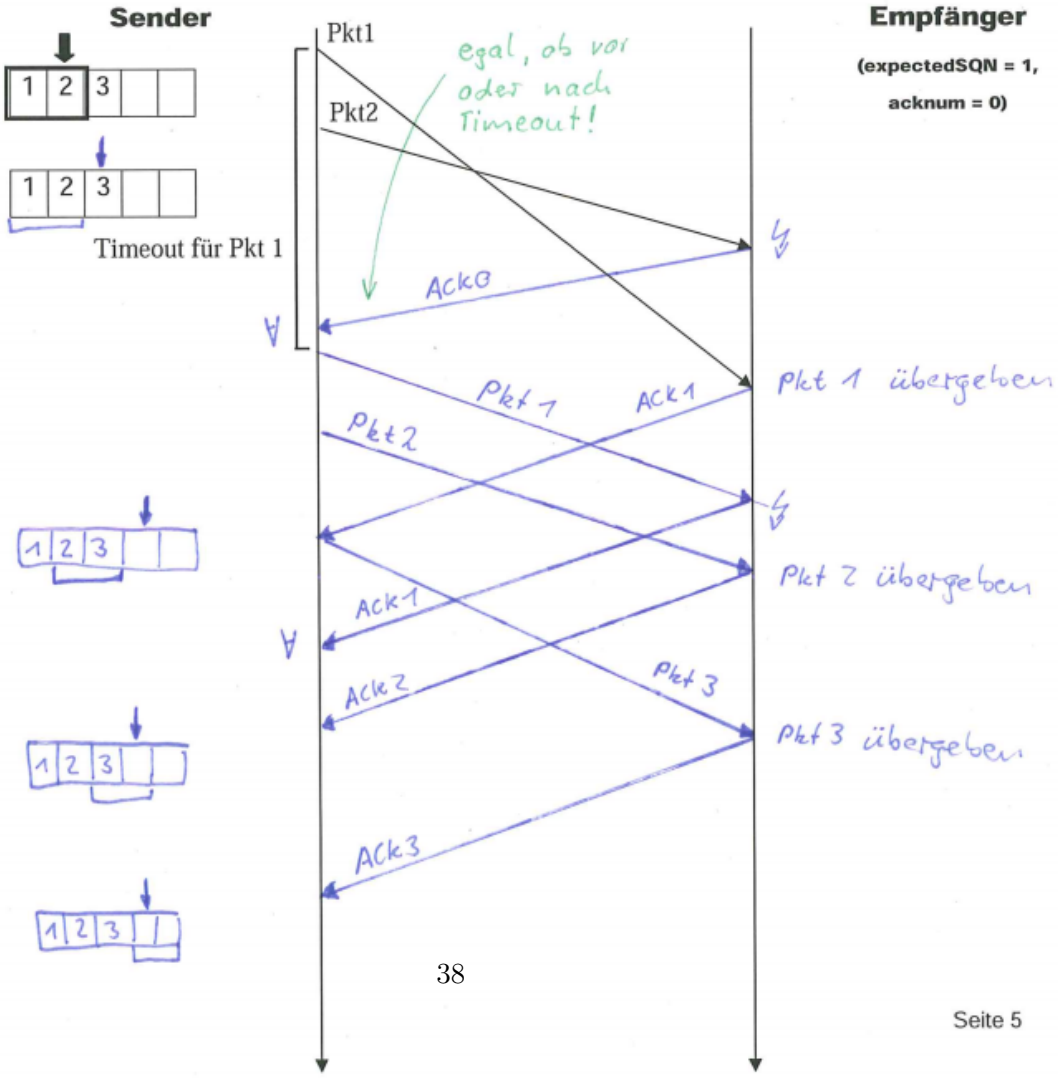
3 Kommunikationsverlauf

3.1 Transportschicht

Aufgabe 2: Transportschicht (25 Punkte)

a) (8 Punkte) Im Diagramm unten ist ein Ablauf des Schiebefensterprotokolls Go-Back-N wie in der Vorlesung dargestellt. Die Fenstergröße beträgt W = 2. Der aktuelle Sendepuffer wird durch das Rechteck gekennzeichnet, der aktuelle Wert von base durch den Pfeil. Dargestellt ist der Zustand nach dem Senden des ersten Pakets. In dem Beispiel sollen drei Pakete verschickt werden. Paket 2 überholt Paket 1 und erreicht den Empfänger vor Paket 1, das nicht mehr innerhalb seines Timeouts (ca. 1.5 RTT) bestätigt werden kann. Führen Sie das Diagramm fort, bis alle drei Pakete beim Sender bestätigt sind. Neben dem Zustand des Sendepuffers nach Senden und Empfangen geben Sie bitte auch an, wann welche Pakete beim Empfänger an die obere Schicht übergeben werden. Es treten keine weiteren Fehler auf, alle weiteren Übertragungszeiten entsprechen der von Paket 2.

next SQN



3.2 Erkennen

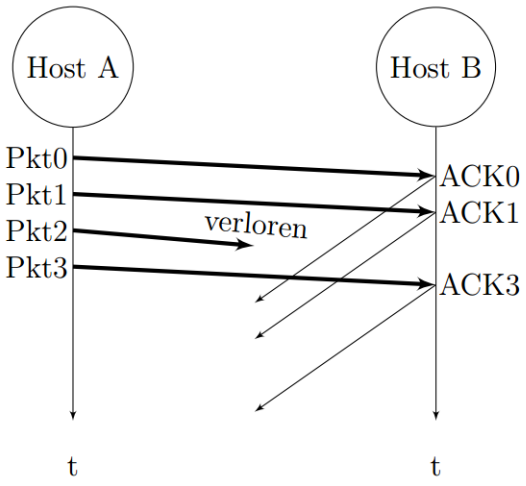


Abbildung 1: Ablaufdiagramm

2 Transportschicht (29 Punkte)

2.1 Schiebefensterprotokoll

Abbildung 1 zeigt den Paketaustausch zwischen 2 Hosts.

Kann auf Grundlage von Abbildung 1 entschieden werden, ob Go-Back-N oder SelectiveRepeat verwendet wird? Begründen Sie Ihre Antwort kurz.
Ja, es wird Selective-Repeat verrwendet, da bei Go-Back-N kein ACK3 gesendet werden würde, bevor nicht eine ACK2 gesendet wurde.
Nehmen Sie an, das in Abbildung 1 gezeigte Schiebefensterprotokoll hat eine Fenstergröße von W = 4 Paketen und der Sequenznummernraum ist 2 Bit breit. Skizzieren Sie die Fensterposition von Host A und Host B kurz nach dem Absenden der letzten Bestätigung (ACK3) und bevor das erste ACK angekommen ist.

| x| normales window segmet
|| x | base segment
| **x** | segment, dass geACKt ist^^

A: ... || 0 | 1 | 2 |3 | ....
B ... | **0** | **1** || 2 |**3** |4| 5|...

daraus wird denke ich auch klar, warum die Sequenznummer net reichen.
| **0** | **1** || 2 |**3** |4| 5|.
in dem beispiel hier bräuchte man schon 6 nummern

Zeigen Sie anhand eines Beispielverlaufs, dass hier ein Sequenznummernfeld von 2 Bit nicht ausreichend ist. Welcher Parameter muss verändert werden um eine eindeutige Übertragung zu gewährleisten?
A sendet Paket 00-11, aber das ACK 00 von B geht verloren. B hat sein Fenster auf die nächsten Pakete 00-11 verschoben. A sendet das alte Paket 00, doch B denkt es sei das neue 00. Sequenznummerfeld also auf 3 Bit erhöhen oder Fenster auf 2 reduzieren.

2.2 TCP Signalisierung (10 Punkte)

In Abbildung 2 baut ein Host A zu einem Host B eine TCP Verbindung auf. Nach einer Anfrage mit einer Größe von 5 B beginnt Host B mit maximaler Geschwindigkeit Daten an Host A zu übertragen. Vervollständigen Sie das in Abbildung 2 gezeigte Ablaufdiagramm durch Eintragen der Sequenznummern (Seq), der Acknowledgment Nummern (Ack) und des Congestion Windows (CW). Gehen Sie davon aus, dass die Puffer immer unendlich Groß sind. Die Fenstergrößen sind nach Erhalten und vor dem Aussenden eines Segments in der Maximum Segment Size (MSS) Einheit anzugeben, wobei gilt: 1 MSS = 10 B. Host A startet mit Seq=50 und Host B mit Seq=0. Es gibt kein Selective Acknowledgment (SACK) und keinen weiteren Timeout. Zu welcher Zeit (t_1 oder t_2) könnte Host B neue Daten übermitteln? Begründen Sie Ihre Antwort.

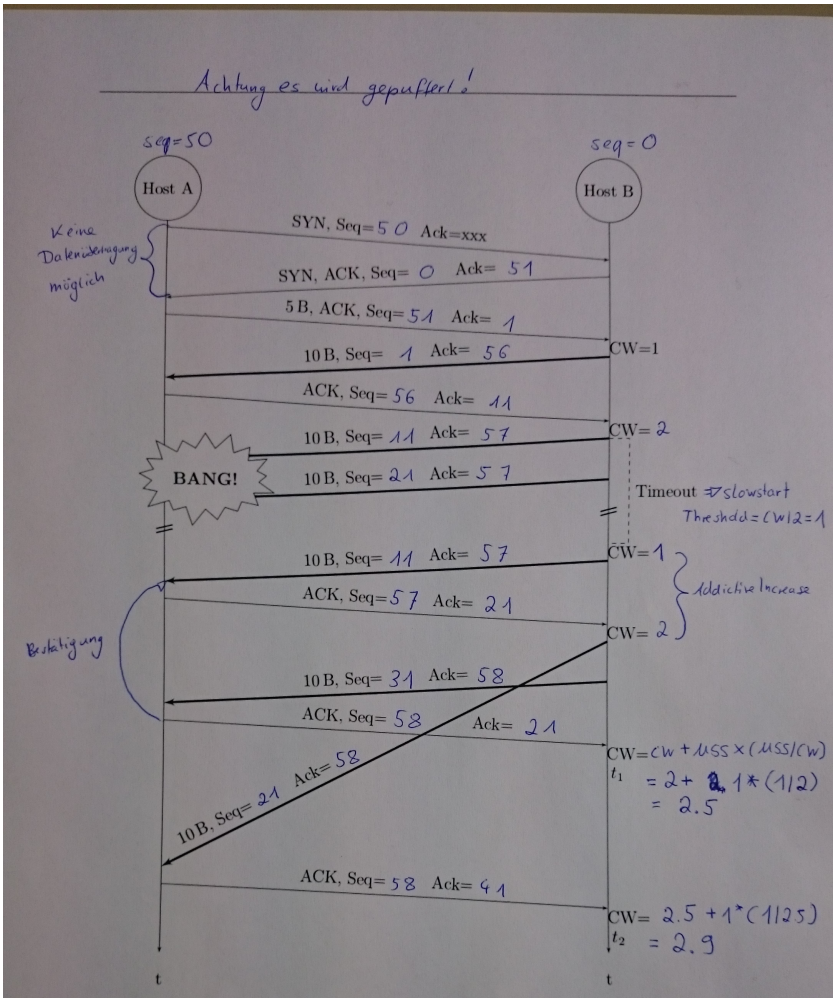


Abbildung 2: Ablaufdiagramm für eine TCP-Verbindung

3.2 Kommunikations-Verlauf [13 Punkte]

Tragen Sie in Abbildung 2 den Ablauf für den Paketaustausch zwischen Host C und Host S auf TCP- und Anwendungsebene ein. Setzen Sie für jedes gesendete Paket die passenden TCP-Flags (keine Sequenz- und Acknowledgement-Nummern) und zeigen Sie an, ob Daten im Paket transportiert werden. Annotieren Sie zusätzlich die übertragenen POP3-Befehle. Vereinfachend wird hier davon ausgegangen, dass die POP3-Befehle eine Größe von 0 Bytes aufweisen (keine Daten). Einzig die übertragene E-Mail hat ein von 0 verschiedenes Datenvolumen. Die Abkürzung CW in Abbildung 2 steht für *Congestion Window* und wird mit 1 *Maximum Segment Size* (MSS) initialisiert (CW = 1 MSS). Nennen Sie weiterhin den Wert des CW nach jedem Paket das bei Host S eingetroffen ist (Berechnung gemäß TCP-Spezifikation aus der Vorlesung). Gehen Sie davon aus, dass das *Advertized Window* von Host C immer groß genug ist. Zeichnen Sie das Diagramm, bis eine E-Mail der Größe $O = 9000 Bytes$ vollständig übertragen ist.

Tipp: Beachten Sie den TCP-Slow-Start.

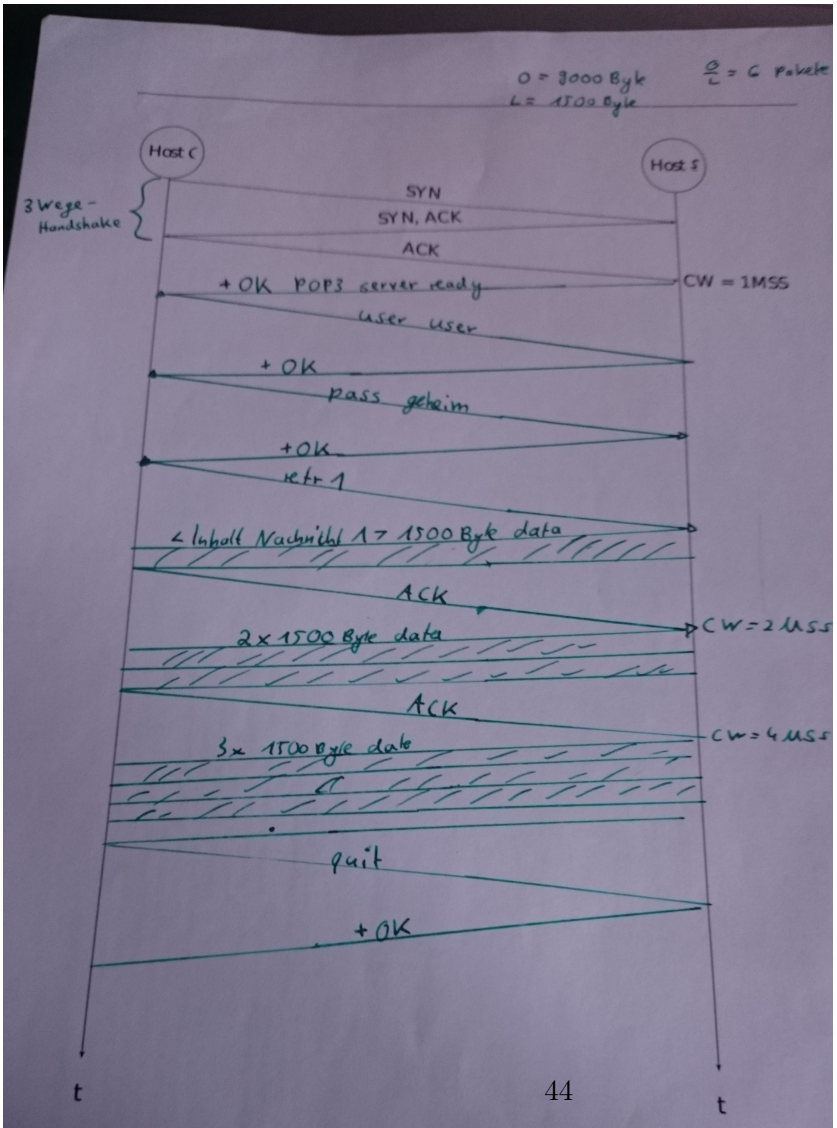


Abbildung 2: Ablaufdiagramm für eine TCP/POP3-Verbindung

Rahmensendedauer ($L = 1000\,\text{bit}$)	$d_{trans} = 1000$ Bitzeiten
Ausbreitungsverzögerung	$d_{AB} = 150$ Bitzeiten $d_{BC} = 300$ Bitzeiten $d_{AC} = 450$ Bitzeiten
Dauer, die der Kanal vor dem tatsächlichen Senden frei sein muss	$t_{sense} = 50$ Bitzeiten
Jam-Signal-Sendedauer	$t_{jam} = 50$ Bitzeiten
Backoff-Konstanten	$K_A = 3$ $K_B = 2$ $K_C = 1$
Backoff-Zeitslot	$\tau = 700$ Bitzeiten

Tabelle 1: Netzwerktopologie

2.2 Carrier Sense Multiple Access (3 + 14 + 3 Punkte)

Nehmen Sie an, dass die drei Hosts aus Abbildung 1 das 1-persistente CSMA/CD-Verfahren für den Medienzugriff verwenden. Weiterhin gelten die Werte aus Tabelle 1.

Host A und Host B wollen zum Zeitpunkt t0 = 0 Bitzeiten gleichzeitig einen Rahmen versenden. Dazu lauschen beide gleichzeitig auf dem Medium und entscheiden sich zum Zeitpunkt t1 = 50 Bitzeiten den Rahmen zu senden. Host C möchte zum Zeitpunkt t2 = 320 Bitzeiten einen Rahmen versenden und beginnt somit zu lauschen. Zeichnen Sie in die Abbildungen 2 und 3 den Rahmenaustausch maßstabsgetreu ein bis alle 3 Pakete erfolgreich versendet wurden. Nehmen Sie (wie in der Übung) vereinfachend an, dass für alle Hosts konstante Backoff-Faktoren gemäß Tabelle 1 gelten.

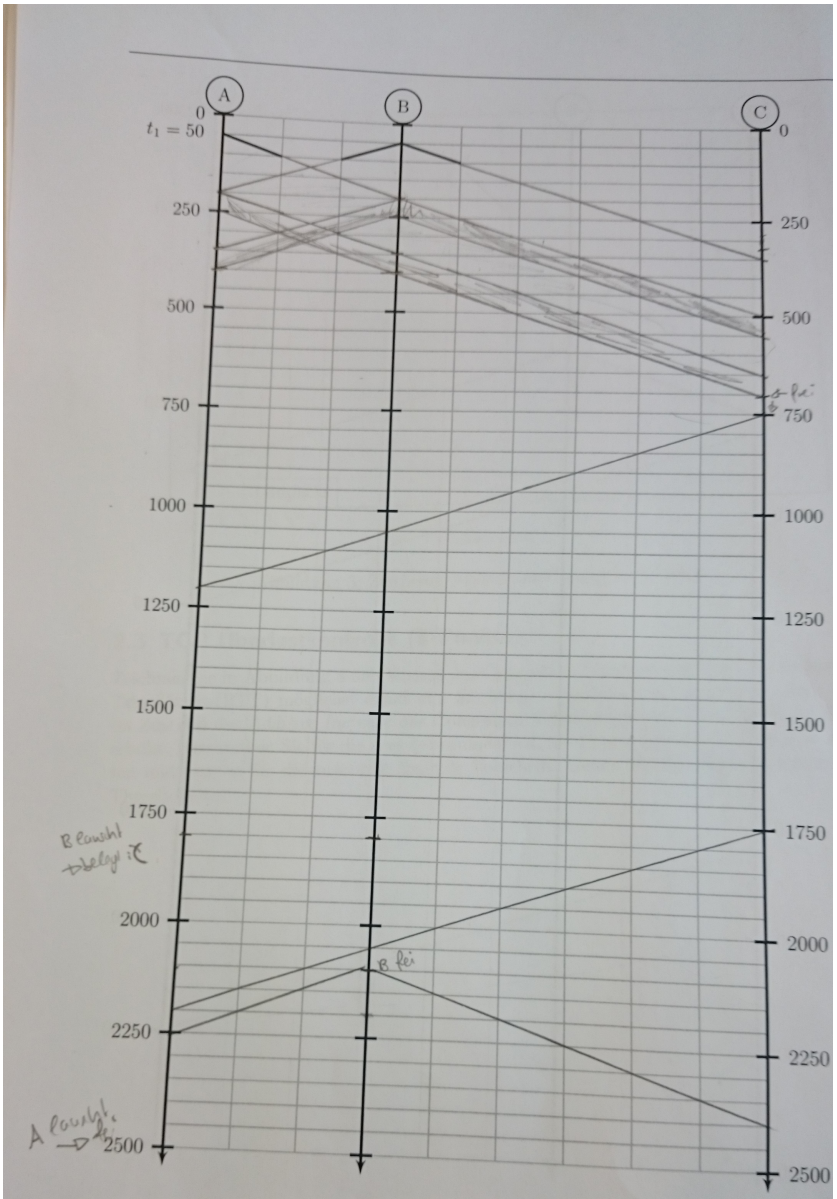


Abbildung 2: Ablaufdiagramm für CSMA/CD

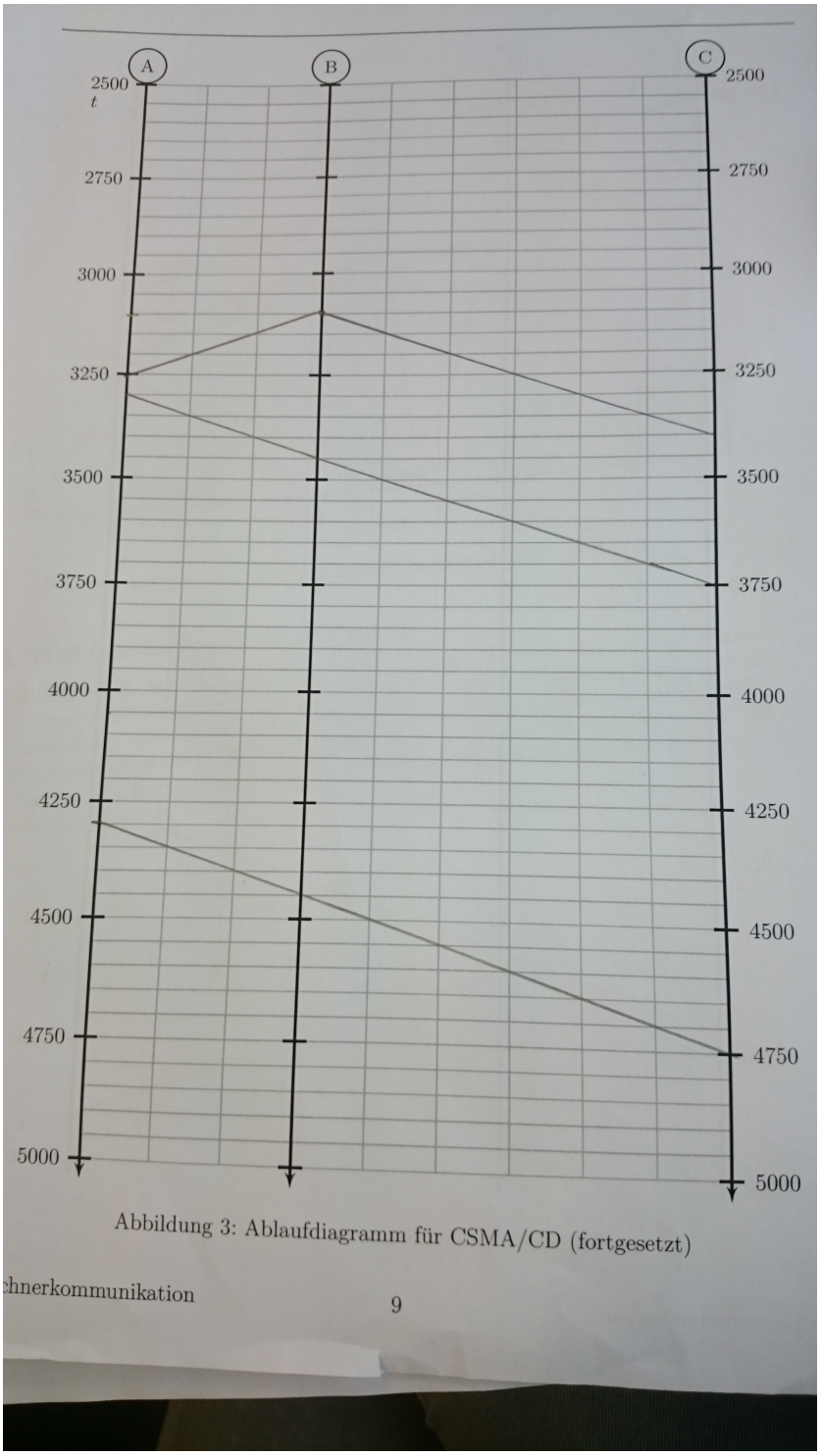


Abbildung 3: Ablaufdiagramm für CSMA/CD (fortgesetzt)

4 Verbindungsablauf

4.1 Tabelle

4.1.1

1 Verbindungsablauf (22 Punkte)

Kurz nach dem Systemstart (es wurde noch keine Pakete versendet oder empfangen) will der Benutzer der Workstation *workstation.lan* auf folgende Webseite zugreifen: *http://server.wan/test.html*. Tragen Sie **alle** Pakete, die die Workstation hierfür im fehlerfreien Fall versendet und empfängt, in Tabelle 2 ein. Der Webserver *server.wan* spricht HTTP/1.1. Die Workstation beendet nach dem vollständigen Empfang die Verbindung. Die Seite besteht aus einem Element mit der Größe L.

Die Abkürzung "Src" steht für die Quell- und die Abkürzung "Dest" für die Ziel-Adresse. Sie können die IP und MAC-Adressen jeweils mit dem letzten Byte abkürzen (z.B. für die MAC-Adresse der Workstation: "F9"). Broadcast-Adressen kürzen Sie mit "FF" ab. Wird ein Feld nicht benötigt so streichen Sie es aus. In der Spalte Protokoll nennen Sie das höchstwertige Protokoll. In die letzte Spalte tragen Sie die wichtigen Optionen und/oder den Payload ein. Eine Pseudonotation ist ausreichend.

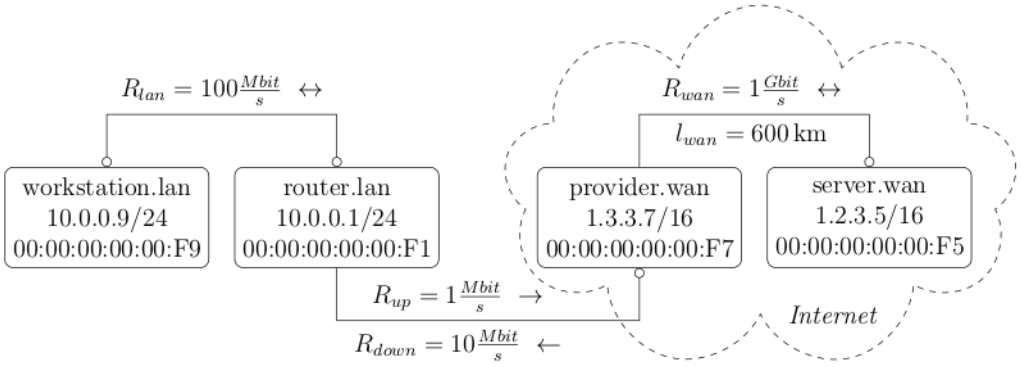


Abbildung 1: Netzwerkkonfiguration

Src MAC	Dest MAC	Src IP	Dest IP	Protokoll	Optionen/Bemerkung/Payload
F9	FF			ARP	Anfrage Mac Adresse Router.Lan(DNS server
F1	F9			ARP	Antwort mit Mac Adresse von Rounter.Lan(DNS
F9	F1	.9	.1	DNS	Anfrage IP von Webseite: server.wan/test.htm
F1	F1	.1	.9	DNS	IP Adresse des Servers
F9	FF			ARP	Anfrage Mac Adresse von server.wan
F1	F9			ARP	Mac Adresse des webservers
F9	F1	.9	.5	TCP	SYN
F1	F9	.5	.9	TCP	SYN ACK
F9	F1	.9	.5	HTTP	Get server.wan/test.html
F1	F9	.5	.9	HTTP	Daten der webseite

4.1.2

2 Verbindungsaufbau (20 Punkte)

Gegeben sei eine Netzwerkkonfiguration gemäß Abbildung 1 (nächste Seite). Sie besteht aus einem *Local Area Network* (LAN) und aus einem *Wide Area Network* (WAN). Beide Netzwerke sind mit einem IP-Router verbunden. Jedes Rechteck bildet einen Teilnehmer ab. In der ersten Zeile steht der Hostname des Teilnehmers, in der zweiten Zeile seine IP-Adresse und in der dritten Zeile seine *Medium Access Control* (MAC) Adresse. Der Router hat doppelt so viele Zeilen, da er über zwei Netzwerkinterfaces verfügt. Die IP-Adressen werden statisch zum Systemstart vergeben. Die Workstation *pc1.lan* kennt die IP-Adresse und den Hostnamen des *Domain Name Servers* (DNS), sowie des IP-Routers. Dem DNS-Server *dns.lan* ist die IP-Adresse des Webservers *www.wan* bekannt.

Kurz nach dem Systemstart will nun der Benutzer der Workstation *pc1.lan* auf folgende Webseite zugreifen: *http://www.wan/test.html*. Tragen sie alle Pakete, die die Workstation hierfür versendet und empfängt, in Tabelle 1 ein.

Die Abkürzung "Src" steht für die Quell- und die Abkürzung "Dest" für die Ziel-Adresse. Sie können die IP- und MAC-Adressen jeweils mit dem letzten Byte abkürzen (z.B. für die MAC-Adresse der Workstation: "F1"). Broadcast-Adressen kürzen sie mit "FF" ab. Wird ein Feld nicht benötigt so streichen Sie es aus. Der Paketverlauf soll bis einschließlich dem HTTP-Request dargestellt werden. In der Spalte Protokoll nennen Sie das höchstwertige Protokoll. In die letzte Spalte tragen Sie die wichtigen Optionen und/oder den Payload ein. Eine Pseudonotation ist ausreichend.

(Abbildung und Tabelle befinden sich auf der nächsten Seite.)

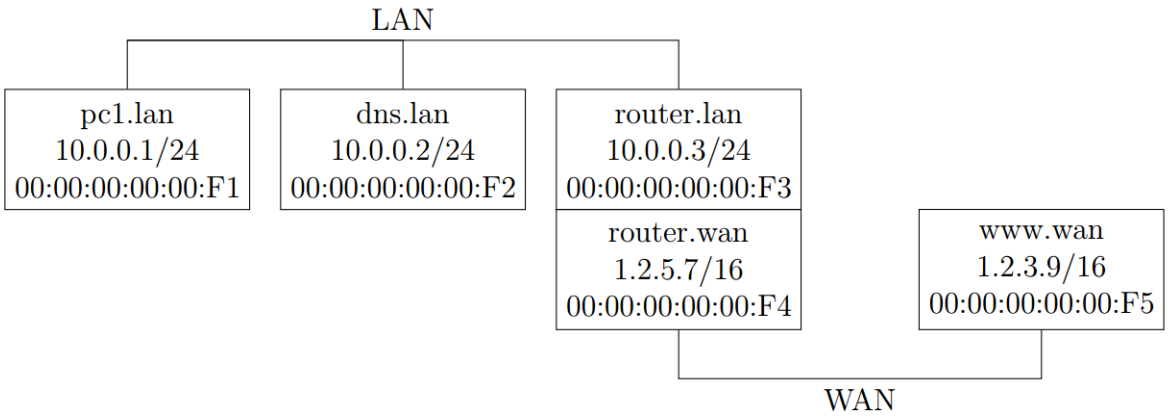


Abbildung 1: Netzwerkkonfiguration

Src MAC	Dest MAC	Src IP	Dest IP	Protokoll	Optionen/Bemerkung/Payload
F1	FF			ARP	Anfrage Mac Adresse dns.lan
F2	F1			ARP	Antwort der Mac Adresse von dns.lan
F1	F2	.2	.1	DNS	Anfrage IP Adresse von Webside
F2	F1	.1	.2	DNS	Antwort IP Adresse von Webside
F1	FF			ARP	Anfrage von Mac Adresse von router.lan
F3	F1			ARP	Antwort der Mac Adresse von Router.lan
F1	F3	.1	.9	TCP	SYN
F4	FF	.9	.1	TCP	SYN ACK
F5	F4	.1	.9	HTTP	Get www.wan/test.html
F4	F5	.9	.1	HTTP	Daten der Webside

4.1.3

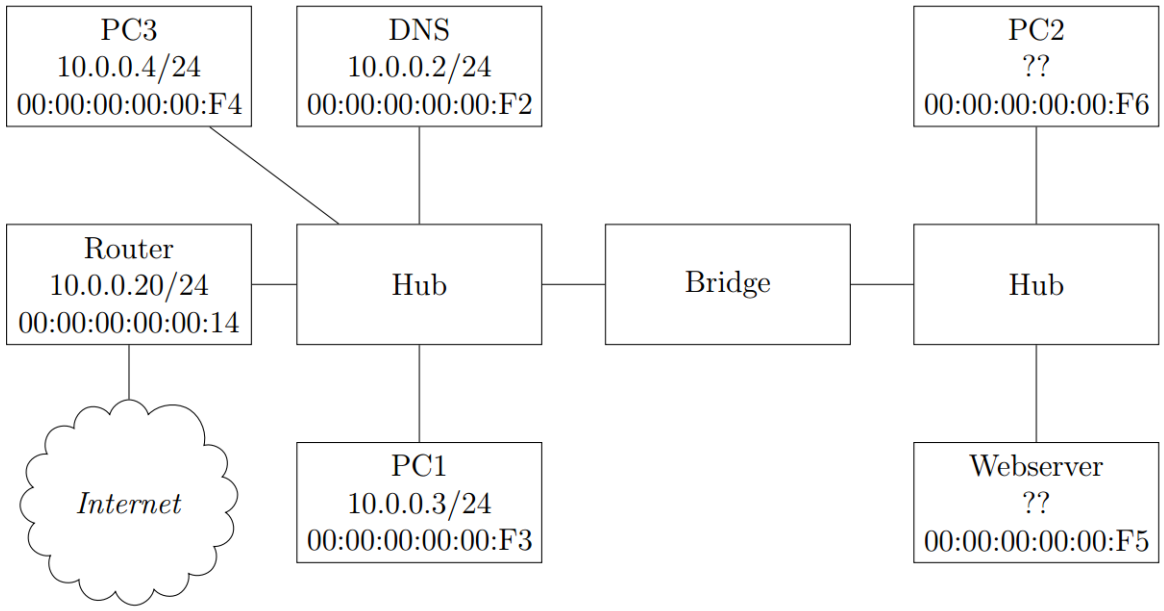


Abbildung 2: Netzwerkkonfiguration

3 IP Netzwerke (29 Punkte gesamt)

Abbildung 2 zeigt ein lokales Netzwerk inklusive einem Router. In der ersten Zeile jedes Hosts steht sein Name, in der zweiten Zeile seine IP Adresse und darunter seine MAC Adresse.

Vergeben Sie sowohl für PC2 als auch für den Webserver eine gültige Adresse, damit jeder Teilnehmer im lokalen Netzwerk jeden anderen erreichen kann.
PC2: 10.0.0.6/24
Webserver 10.0.0.5/24

PC2 sendet einen Ethernet-Rahmen an den Webserver. Zur exakt gleichen Zeit sendet PC1 einen Ethernet-Rahmen an den Router. Kollidieren die beiden Rahmen? Begründen Sie Ihre Antwort kurz
Ja es kommt zur Kollision im rechten Hub, die Brdige hat damit nichts zu tun, Hubs sind primitive Wesen auf dem Physical Layer, beherschen jedoch meistens CSMA/CD.
PC1 sendet einen Ethernet-Rahmen an den Webserver. Zur exakt gleichen Zeit sendet PC3 einen Ethernet-Rahmen an den Router. Kollidieren die beiden Rahmen? Begründen Sie Ihre Antwort kurz.
Ja es kommt zur Kollision im linken Hub, die Bridige hat damit nichts zu tun, da sie ja nicht verhindert dass sich die Pakete auf dem Weg rum Router im Linken Hub treffen. Die Bridge wuerde verhindern, dass es eine Kollision gibt wenn PC1 an PC3 sendet und PC2 an den Webserver (fiktives Beispiel).
PC1 versendet eine ARP-Anfrage um die IP-Adresse des Routers zu ermitteln. Empfängt PC2 diesen Rahmen? Begründen Sie Ihre Antwort kurz. ja.Ein Broadcast wird stets in alle Teilnetze übertragen.

e) Ein Browser auf PC1 möchte ein HTML-Dokument (index.html) mit einem eingebetteten Bild (pic.jpg) von dem Webserver laden. Das Dokument und das Bild passen jeweils in ein TCP-Segment. Sowohl der Webserver als auch PC1 verwenden beide HTTP/1.1 mit persistenten Verbindungen. PC1 kennt zunächst nicht die IP-Adresse des Webservers und muss diese erfragen.

Vervollständigen Sie in Tabelle 1 den zeitlichen Verlauf der ausgetauschten Pakete auf Transport- und Anwendungsschicht, bis alle Verbindungen wieder geschlossen sind. Geben Sie im Optionsfeld bei TCP Segmenten immer an, welche Flags (SYN, FIN und/oder ACK) gesetzt sind. (8 Punkte)

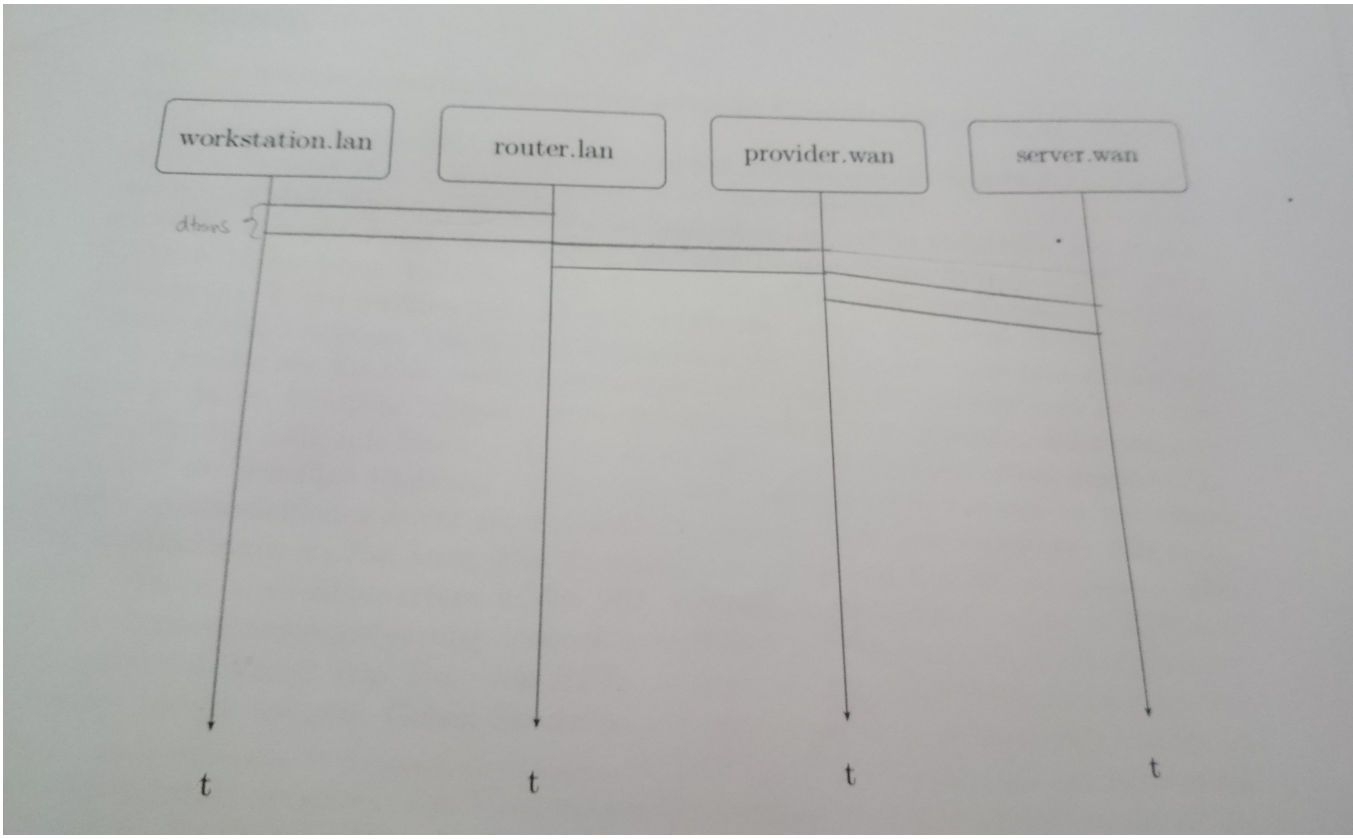
Quellname	Zielname	Protokol(Transport)	Protokol(Anwendung)	Optionen/Bemerkung/Payload
DNS	PC1	UDP	DNS	10.0.0.6
PC1	Webserver	TCP		SYN
Webserver	PC1	TCP		SYN,ACK
PC1	Webserver	TCP	HTTP	ACK, GET index.html
Webserver	PC1	TCP	HTTP	index.
PC1	Webserver	TCP	HTTP	ACK, GET pic.
Webserver	PC1	TCP	HTTP	pic.jpg
PC1	Webserver	TCP		ACK, FIN
Webserver	PC1	TCP	HTTP	ACK
Webserver	PC1	TCP		FIN
PC1	Webserver	TCP	HTTP	ACK

4.2 Diagramm

4.2.1 Store and Forward

Tragen sie folgenden Ablauf in Abbildung 2 ein:
dtrans: jeder Link

dprop: nur letzter Link
(Achtung vereinfachte Aufgabe)



4.2.2 Cut-through

2.4 Paketaustausch (5 Punkte)

Tragen Sie einen Paketverlauf für das *Cut-through* Verfahren in Abbildung 4 ein. Die Zusammensetzung des Paket (einzelne Header und Nutzdaten) muss klar erkennbar sein.

(Achtung vereinfachte Aufgabe)

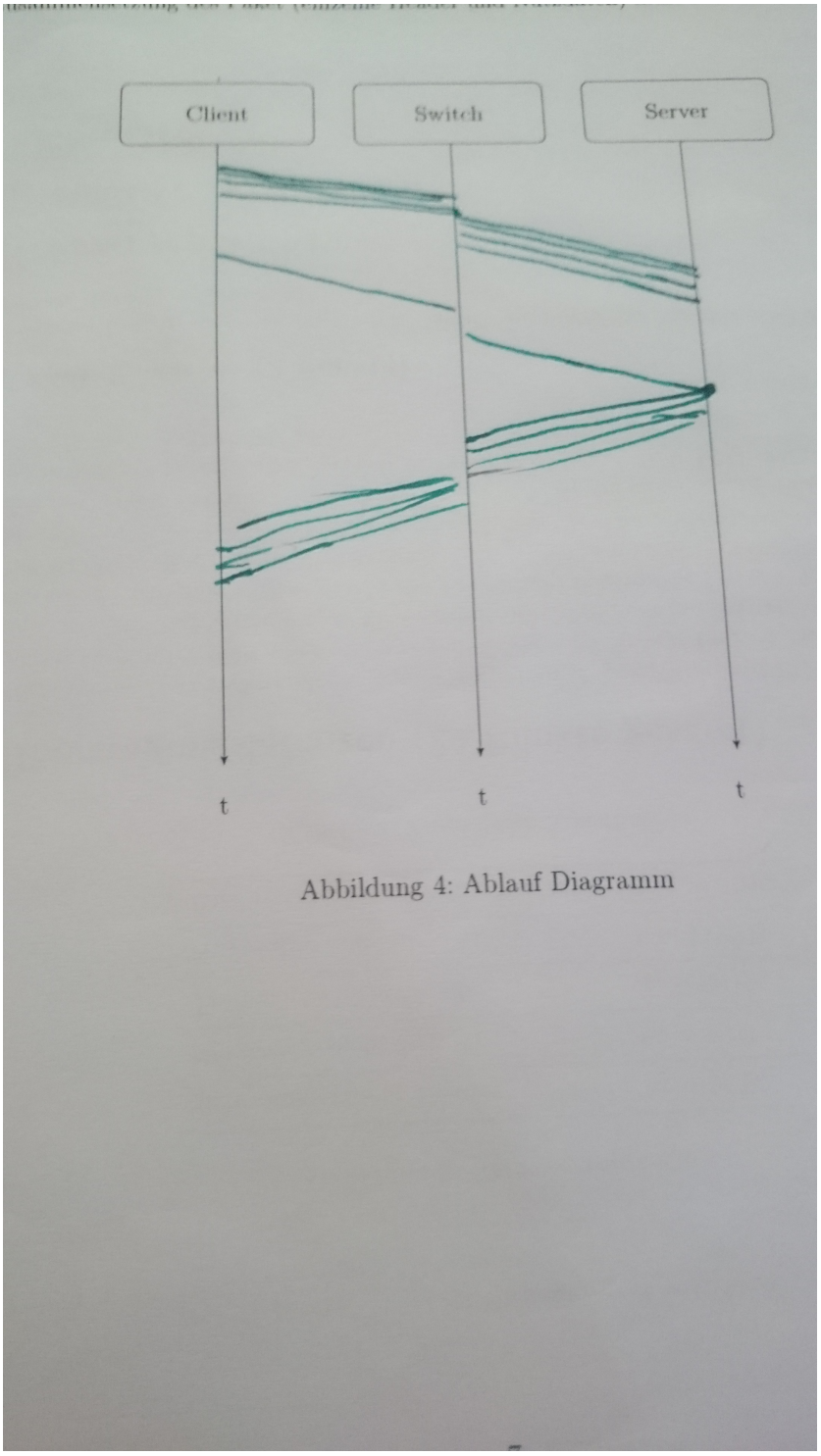


Abbildung 4: Ablauf Diagramm

5 Programmieraufgaben

5.1 Statechart

5.1.1

---

4 Programm (20 Punkte)

Implementieren Sie in JAVA blockierungsfrei die Senderseite eines zuverlässigen Transportprotokolls, dessen Verhalten in Abbildung 4 gegeben ist. Es werden nur unidirektional Nutzdaten übermittelt. Leiten Sie die Klasse *SendingHost* von der Klasse *CommonHost* (Listing 2) ab und implementieren Sie unter Berücksichtigung folgender Aspekte das Verhalten (die *abstract*-Methoden müssen implementiert werden):

- Die Methode *startTimer()* startet einen Timer, der nach Ablauf einer fest eingestellten Zeit die Methode *timeout()* aufruft, wenn die Methode *stopTimer()* nicht aufgerufen wird.
- Die Methode *fromAbove(Pkt)* wird aufgerufen, wenn die darüber liegende Schicht ein Paket versendet. Sollte eine Übertragung gerade nicht möglich sein, so liefert sie ein *false* zurück. Sonst *true*.
- Die Methode *transmit(Pkt)* stellt den eigentlich Sendevorgang eines Pakets dar.
- Die Methode *receive(Ack)* wird aufgerufen, nachdem ein Paket empfangen wurde.
- Die Methode *bitError(Pkt)* überprüft, ob das Paket Fehler enthält. *true = fehlerhaft*
- Sie dürfen (wenn benötigt) andere Methoden oder Variablen hinzufügen, um das gewünschte Verhalten zu erzielen.
- Sie benötigen **keine** Adressierung.
- Zum Datenaustausch zwischen Ihren eigenen Routinen und dem Kommunikationspartner werden Pakete gemäß Listing 1 verwendet.
- Die Pakete können in einem *großem Array* zwischengespeichert werden. Sie müssen sich nicht um den Überlauf des Arrays kümmern.

Listing 1: Paket zum Datenaustausch

```
1  import java.*;
2
3  public class Pkt
4  {
5          int seqnum;
6          int acknum;
7          int checksum;
8          byte payload[] = new byte[20];
9  }
```

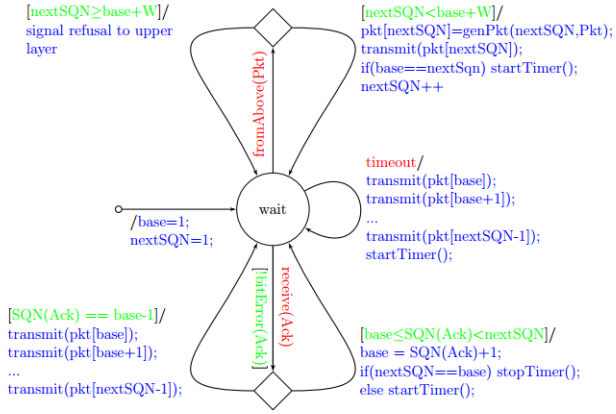


Abbildung 4: Zustandsdiagramm

Listing 2: Abstrakte Klasse CommonHost

```
1  import java.*;
2
3  public abstract class CommonHost
4  {
5          static const int W = 8;
6
7          void startTimer() {...}          //startet den Timer
8          void stopTimer() {...}           //stoppt den Timer
9          void transmit(Pkt pkt) {...}
10         int checkSum(Pkt pkt) {...}
11         boolean bitError(Pkt pkt) {...}
12         abstract boolean fromAbove(Pkt pkt);
13         abstract void receive(Pkt pkt);
14         abstract void timeout();
15 }
```

```
1
2  public class SendingHost extends CommonHost {
3
4          private int base = 1;
5          private int nextSQN = 1;
6          private Pkt buffer[] = new Pkt[Integer.MAX_VALUE];
7
8          private Pkt genPkt(int sqn, Pkt p) {
9                  p.seqnum = sqn;
10                 p.checksum = checkSum(p);
11                 return p;
12         }
13
14         public boolean fromAbove(Pkt pkt) {
15                 if(pkt.seqnum >= base + W) {
16                         return false;
17                 }
18
19                 buffer[nextSQN] = genPkt(nextSQN, Pkt);
20                 transmit(buffer[nextSQN]);
21                 if(base == nextSQN) {
22                         startTimer();
23                 }
24                 nextSQN++;
25
26                 return true;
27         }
28
29         public void receive(Pkt pkt) {
30                 if(bitError(pkt)) {
31                         return;
32                 }
33
34                 if(pkt.seqnum == base - 1) {
35                         for(int i = base; i < nextSQN; i++)) {
36                                 transmit(pkt[i]);
37                         }
38                         return;
39                 }
40
41                 if(base <= pkt.seqnum && pkt.seqnum < nextSQN) {
42                         base = pkt.seqnum + 1;
43                         if(nextSQN == base) {
44                                 stopTimer();
45                         } else {
46                                 startTimer();
47                         }
48                         return;
49                 }
50         }
51
52         public void timeout() {
```

```
53                  for(int i = base; i < nextSQN; i++)) {
54                          transmit(buffer[i]);
55                  }
56                  startTimer();
57          }
58  }
```

5.1.2

3 TCP-FIN: Statechart-Programmierung (22 Punkte)

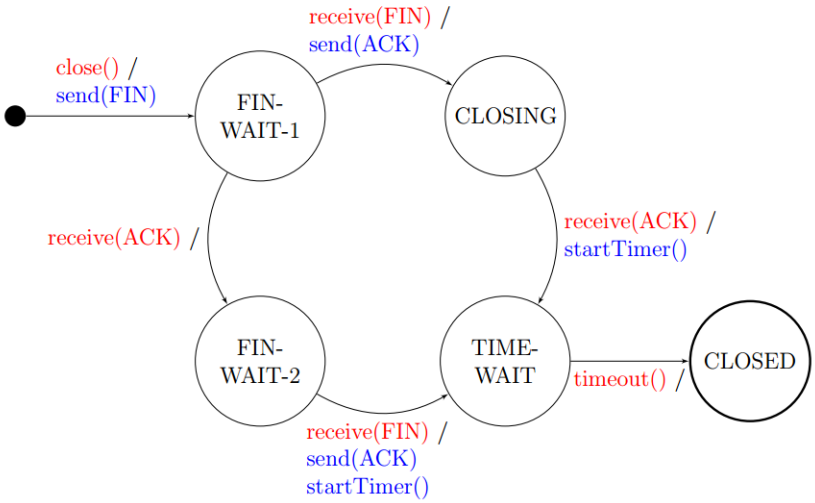


Abbildung 3: TCP-FIN Initiator Verhalten

Abbildung 3 zeigt das Verhalten für einen TCP-Verbindungsabbau auf der Seite des Initiators im fehlerfreien Fall. Realisieren Sie das gezeigte Verhalten blockierungsfrei in der Klasse *TCPFIN*. Leiten Sie die Klasse *TCPFIN* von *Base* ab und verwenden Sie die in *Base* bereit gestellten Methoden. Führen Sie eine Zustandsvariable ein, die klar die Zustände aus Abbildung 3 widerspiegelt. Bei Fehlern werfen einige Methoden eine *WrongStateException* bzw. eine *IOException*. Diese Exceptions müssen nicht behandelt werden und führen zum Abbruch des Programms. Ein fehlerfreies Kompilieren muss dennoch gewährleistet sein. Die *WrongStateException* wird von der Methode *timeout()* geworfen, wenn diese aufgerufen wird ohne dass sich das Programm im Zustand *TIME-WAIT* befindet. Beachten Sie die Kommentare von *Base* in Listing 1.

Listing 1: Abstrakte Klasse Base

```
1  import java.io.IOException;
2
3  public abstract class Base throws Exception {
4
5          /* Flags */
6          final int FIN = 0;
7          final int ACK = 1;
8
9          /* Exception fuer einen falschen Systemzustand */
10         class WrongStateException extends Exception {...}
11
12         /* Sendet ein Flag zum Kommunikationspartner */
13         void send(int flag) throws IOException {...}
14
15         /* Startet einen internen Timer */
16         void startTimer() {...}
17
18
19         /**
20          * Folgendes muss in TCPFIN implementiert werden
21          **/
22
23         /* Die Applikation moechte die Verbindung beenden */
24         abstract void close(void) throws IOException;
25
26         /* Empfaengt ein Flag vom Kommunikationspartner */
27         abstract void receive(int flag) throws IOException;
28
29         /* Wird nach Ablauf des Timers aufgerufen */
30         abstract void timeout(void) throws WrongStateException;
31 }
```

```
public class TCPFIN extends Base{
    public enum STATE ={FIN-WAIT-1, CLOSING, TIME-WAIT, FIN-WAIT-2, CLOSED,
     init};
    private volatile STATE state = init;

    public void timeout() throws WrongStateException{
           if(state!=TIME-WAIT{
              throw new WrongStateException();
           }
           state = CLOSED;
    }

    public void close() throws IOException{
           if(!state==init)
              throw new IOException();
           send(FIN);
           state =FIN-WAIT-1;
    }

    public void receive(int flag) throws IOE{
           switch(state){
           case FIN-WAIT-1:
              if(flag == FIN){
                 state = CLOSING;
                 send(ACK);
              }else{
                 state=FIN-WAIT-2;
              }
           break;
           case CLOSING:
              if(flag==ACK){
                           state =TIME_WAIT;
                           startTimer();
                 }else{
                           throw new IOExceptio();
              }
           break;
           case FIN-WAIT-2:
              if(flag==FIN){
                    send(ACK);
                    state =TIME-WAIT;
                    startTimer();
              }else{
                    throw new IOException();
                 }
           break;
           default:
              throw new IOExcpetion();
           }
    }
}
```

5.1.3

3.3 Programm (20 Punkte)

Realisieren Sie das in Abbildung 2 (nächste Seite) gezeigte Verhalten in JAVA. Jeder Teilnehmer kann sowohl senden als auch empfangen. Leiten Sie die Klasse *BehaviorImpl* von der Klasse *Behavior* (Listing 1) ab und implementieren Sie unter Berücksichtigung folgender Aspekte das Verhalten (die *abstract*-Methoden müssen implementiert werden):

- Die Methode *startTimer(ms)* startet einen Timer, der nach *ms* Millisekunden die Methode *timeout()* aufruft, wenn die Methode *stopTimer()* nicht aufgerufen wird.
- Die Methode *fromAbove(data)* wird aufgerufen, wenn die darüber liegende Schicht Daten versendet.
- Die Methode *toAbove(data)* übergibt die empfangenen Daten an die höhere Schicht.
- Die Methode *tansmit(data)* stellt den eigentlich Sendevorgang eines Rahmens dar. Sie soll zurückkehren, sobald das Senden beendet ist.
- Die Methode *receive(data)* wird aufgerufen, nachdem ein Rahmen empfangen wurde.
- Sie dürfen (wenn benötigt) andere Methoden oder Variablen hinzufügen, um das gewünschte Verhalten zu erzielen.
- Zufallszahlen zwischen 0 (inklusive) und i (exklusive) können mit *rnd.nextInt(i)* generiert werden.
- Implementieren Sie das Zustandsdiagramm blockierungsfrei.
- Sollte die Methode *fromAbove(data)* von der darüber liegenden Schicht aufgerufen werden, während ein anderer Sendevorgang noch nicht abgeschlossen ist, so soll eine Exception geworfen werden: *throw new NotReadyException*.
- Sie benötigen **keine** Adressierung oder Redundanzprüfung.
- Rahmen, die lediglich die Zeichenkette „K“ beinhalten, sollen als Bestätigungspaket (ACK) angesehen werden. (*a.equals(b)* → *true*, wenn a und b die gleiche Zeichenkette beinhalten.)

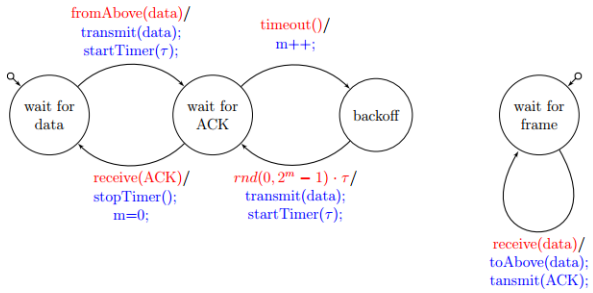


Abbildung 2: Zustandsdiagramm

Listing 1: Abstrakte Klasse Behavior

```
1  import java.*;
2
3  public abstract class Behavior
4  {
5          final int       TAU = 100;        //in Millisekunden
6          final String    ACK = "K";
7
8          int m = 0;                        //Anzahl Fehlversuche
9          Random rnd = new Random();
10
11         public class NotReadyException extends Exception {...}
12         void startTimer(int ms) {...}     //startet den Timer
13         void stopTimer() {...}            //stopt den Timer
14         void toAbove(String data) {...}
15         void transmit(String data) {...}
16
17         abstract void fromAbove(String data) throws
               NotReadyException;
18         abstract void receive(String data);
19         abstract void timeout();
20 }
```

```
1
2  public class BehaviorImpl extends Behavior {
3
4          private final int WAIT_FOR_DATA = 1;
5          private final int WAIT_FOR_ACK = 2;
6          private final int BACKOFF = 3;
7
8          private int state = WAIT_FOR_DATA;
9
10         private String d;
11
12         public void fromAbove(String data) throws NotReadyException {
13                 if(state == WAIT_FOR_DATA) {
14                         d = data;
15                         state = WAIT_FOR_ACK;
16                         transmit(d);
17                         startTimer(TAU);
18                 } else throw new NotReadyException("nudel");
19         }
20    public void receive(String data) {
21                 if(data.equals(ACK) && state == WAIT_FOR_ACK) {
22                         state = WAIT_FOR_DATA;
23                         stopTimer();
24                         m = 0;
25                 } else {
26                         toAbove(data);
27                         transmit(ACK);
28                 }
29         }
30
31         public void timeout() {
32                 if(state == WAIT_FOR_DATA) {
33                         m++;
34                         state = BACKOFF;
35                         startTimer(rnd.nextInt(Math.pow(2, m)));
36                         return;
37                 }
38
39                 if(state == BACKOFF) {
40                         transmit(d);
41                         startTimer(TAU);
42                         state = WAIT_FOR_ACK;
43                         return;
44                 }
45         }
46 }
```

5.2 ALOHA-Zustandsdiagramm

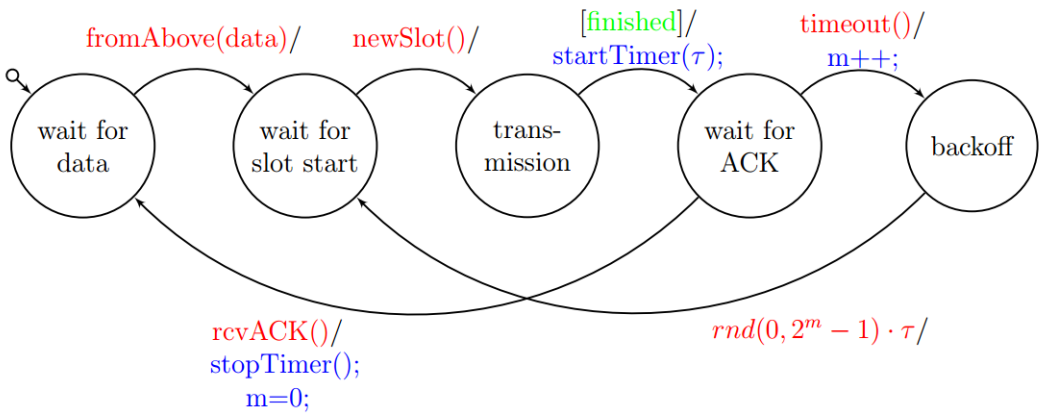


Abbildung 4: *Slotted ALOHA* - Zustandsdiagramm

3 Slotted ALOHA (25 Punkte)

Abbildung 4 zeigt die Senderseite einer *Slotted-Aloha*-basierten Übertragung. Realisieren Sie das gezeigte Verhalten blockierungsfrei in der Klasse *Saloha*. Leiten Sie die Klasse *Saloha* von *Base* ab und verwenden Sie die in *Base* bereitgestellten Methoden. Führen Sie eine Zustandsvariable ein, die klar die Zustände aus Abbildung 4 widerspiegelt. Bei Fehlern werfen einige Methoden eine *WrongStateException* bzw. eine *IOException*. Diese Exceptions müssen nicht behandelt werden und führen zum Abbruch des Programms. Ein fehlerfreies Kompilieren muss dennoch gewährleistet sein. Die Methode *fromAbove()* soll eine Exception werfen wenn sie aufgerufen wird ohne, dass das Zustandsdiagramm dies vorsieht. Beachten Sie die Kommentare von *Base* in Listing 1.

Listing 1: Abstrakte Klasse Base

```
1  import java.io.IOException;
2
3  public abstract class Base throws Exception
4  {
5
6          class Packet {...}
7          final int TAU = 10; //Slotzeit
8
9          /* Exception fuer einen falschen Systemzustand */
10         class WrongStateException extends Exception {...}
11
12         /* Sendet sofort ein Paket zum Kommunikationspartner */
13         void send(Packet pkt) throws IOException  {...}
14
15         /* Startet den Timer der in s Millisekunden ablaeuft */
16         void startTimer(int s) {...}
17
18         /* Stopt den Timer */
19         void stopTimer() {...}
20
21         /* Liefert eine Zufallszahl im Bereich [b,e) zurueck */
22         int rnd(int b, int e) {...}
23
24
25         /**
26          * Folgendes muss in Saloha implementiert werden
27          **/
28
29         /* Wird aufgerufen wenn Daten verschickt werden sollen*/
30         abstract void fromAbove(Packet pkt) throws
               WrongStateException;
31
32         /* Wird nach Ablauf des Timers aufgerufen */
33         abstract void timeout();
34
35         /* Wird ZYKLISCH zu JEDEM Slotbegin aufgerufen */
36         abstract void newSlot() throws IOException;
37
38         /* Wird aufgerufen wenn ein ACK empfangen wird */
39         abstract void rcvACK();
40 }
```

```
public class Saloah extends Base {

        private final int WAIT_FOR_DATA = 1;
        private final int WAIT_FOR_SLOT_START = 2;
        private final int TRANSMISSION = 3;
        private final int WAIT_FOR_ACK = 4;
        private final int BACKOFF = 5;

        private int state = WAIT_FOR_DATA;

        private Packet p;

        public void fromAbove(Packet pkt) throws WrongStateException {
                if(state != WAIT_FOR_DATA) {
                        throw new WrongStateException("nudel");
                }
                state = WAIT_FOR_SLOT_START;
                p = pkt;
        }

        public void timeout() {
                if(state == WAIT_FOR_ACK) {
                        state = BACKOFF;
                        m++;
                        startTimer(rnd(0, Math.pow(2, m)));
                        return;
                }
                if(state == BACKOFF) {
                        state = WAIT_FOR_SLOT_START;
                        return;
                }
        }

        public void newSlot() throws IOException {
                if(state == WAIT_FOR_SLOT_START) {
                        state = TRANSMISSION;

                        send(p);
                        // ... Paket wird gesendet ...

                        state = WAIT_FOR_ACK;
                        startTimer(TAU);
                }
        }

        public void revACK() {
                if(state == WAIT_FOR_ACK) {
                        stopTimer();
                        m = 0;
                        state = WAIT_FOR_DATA;
                }
        }
```

```
53 }
```

3 Mail-Client (25 Punkte)

Implementieren Sie ein **vollständiges** Programm in JAVA, dass die Anzahl der auf einem POP3-Server befindlichen E-Mails erfragt und diese Information per Mail weiterleitet. Die Rückantworten der beteiligten Server sollen überprüft werden.

- Der POP3-Server liegt an `mail.rk.org`:110 (Benutzer: `rk`, Passwort: `rK!23`)
- Als SMTP-Server verwenden Sie: `cipmail.cs.fau.de`:25
- Die Nachricht an `mailCheck@cs.fau.de` soll lauten: `RK hat XX E-Mails`. Wobei `XX` die Anzahl der auf dem RK-Konto befindlichen Mails darstellt. Die E-Mail-Adresse des Absenders soll lauten: `status@rk.org`.

Um die Korrektheit der Rückantworten der Server zu überprüfen können Sie den dreistelligen numerischen Code (z.B. 220, 250, ...) bzw. `+OK`, der zu Beginn jeder Zeile des Servers gesendet wird, verwenden. Kommt eine falsche Antwort von einem der Server, solll die Methode `exit()` aufgerufen werden.Vervollständigen Sie den Code in Listing 3, indem Sie die beiden Methoden `void sendEmail(int count)` und `int getEmailsCount()` implementieren. Geöffnete Verbindungen müssen wieder geschlossen werden. Exceptions müssen berücksichtigt, aber nicht behandelt werden.

Tipp: Beachten Sie die beigefügten Beispiele in Listing 1 und Listing 2.

Listing 1: POP3 Beispiel

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on
C: list
S: 1 498
S: 2 912
S: 3 812
S: .
C: retr 1
S: <Inhalt Nachricht 1>
C: dele 1
S: +OK message marked for delete
C: quit
S: +OK POP3 server signing off
```

Listing 2: SMTP Beispiel

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: How are you?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

```
1
2  import java.io.*;
3  import java.net.*;
4
5  public class RKMailClient {
6
7          private final static String CRLF = "\r\n";
8          private static Socket socket;
9          private static BufferedReader rx;
10         private static BufferedWriter tx;
11
12         public static void main(String[] args) throws Exception {
13                 sendEmail(getEmailsCount());
14         }
15
16         public static void exit() {
17                 System.out.println("Fehler aufgetreten");
18                 System.exit(-1);
19         }
20
21         private static void check(String s) {
22                 if(!rx.readLine().startsWith(s)) {
23                         tx.close();
24                         rx.close();
25                         socket.close();
26                         exit();
27                 }
28         }
29
30         private static int getEmailsCount() throws Exception {
31                 socket = new Socket("mail.rk.org", 110);
32                 rx = new BufferedReader(new InputStreamReader(socket.
       getInputStream()));
33                 tx = new BufferedWriter(new OutputStreamWriter(socket.
       getOutputStream()));
34                 check("+OK");
35                 tx.write("user rk" + CRLF);
36                 check("+OK");
37                 tx.write("pass rK!23" + CRLF);
38                 check("+OK");
39                 tx.write("list" + CRLF);
40
41                 int count = 0;
42                 boolean fin = false;
43                 do {
44                         if(rx.readLine().startsWith(".")) {
45                                 fin = true;
46                         } else {
47                                 count++;
48                         }
49                 } while(!fin);
50
51                 tx.write("quit" + CRLF);
```

```
52                  check("+OK");
53
54                  tx.close();
55                  rx.close();
56                  socket.close();
57                                  return count;
58          }
59
60          private static void sendEmail(int count) throws Exception {
61                  socket = new Socket("cipmail.cs.fau.de", 25);
62                  rx = new BufferedReader(new InputStreamReader(socket.
    getInputStream()));
63                  tx = new BufferedWriter(new OutputStreamWriter(socket.
    getOutputStream()));
64
65                  check("220");
66                  tx.write("HELO rk.org" + CRLF);
67                  check("250");
68                  tx.write("MAIL FROM: <status@rk.org>" + CRLF);
69                  check("250");
70                  tx.write("RCPT TO: <mailCheck.cs.fau.de>" + CRLF);
71                  check("250");
72                  tx.write("DATA" + CRLF);
73                  check("354");
74                  tx.write("RK hat " + count + " E-Mails." + CRLF + "." +
    CRLF);
75                  check("250");
76                  tx.write("QUIT" + CRLF);
77                  check("221");
78
79                  tx.close();
80                  rx.close();
81                  socket.close();
82          }
83  }
```

5.4 POP3-Serverprogrammieung

3 POP3-Server-Programmierung (30 Punkte)

Programmieren Sie einen vereinfachten POP3-Server in JAVA, der auf eine Verbindung von einem POP3-Client wartet, die Authentifizierung des Benutzers überprüft und Anfragen (STAT, LIST, QUIT) des Clients beantworten kann. Jede Antwort des Servers beginnt mit +OK oder –ERR, worauf weitere Informationen bis zur Steuerzeichensequenz „$\backslash r \backslash n$“ folgen können. Bei LIST gibt es eine mehrzeilige Antwort, deren Ende durch eine Zeile, die nur einen Punkt enthält, markiert wird.

Realisieren Sie das in Abbildung 5 gezeigte Verhalten in der *run()*-Methode der Klasse *Pop3Server*. Es ist dabei anzunehemen, dass gleichzeitig nur ein Client bedient wird und dass dieser nur die im Statechart spezifizierten Befehle in der angegebenen Reihenfolge sendet. Leiten Sie die Klasse *Pop3Server* von *Base* ab und nehmen Sie an, dass die Methoden für die Kommunikation mit dem Client, Authentifizierung und Ausführung von POP3-Befehlen in *Base* bereits implementiert sind. Bei Fehler werfen die Methoden eine *POP3Exception* bzw. eine *IOException*. Die *IOExceptions* müssen dabei nicht behandelt werden und führen zum Abbruch des Programms. Die *POP3Exceptions* müssen dagegen abgefangen und mit einer –ERR Nachricht beantwortet werden. Beachten Sie den unten begefügten Beispielablauf in Listing 1 sowie die Kommentare von *Base* in Listing 2.

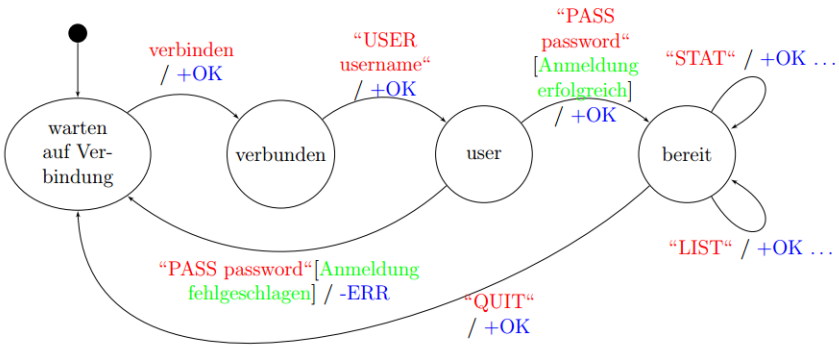


Abbildung 5: POP3-Serververhalten

Listing 1: Beispielablauf POP3-Session

```
S: (wartet auf Verbindung)
C: (oeffnet eine Verbindung)
S: +OK example.com POP3-Server
C: USER username
S: +OK Please enter password
C: PASS passwort_in_klartext
S: +OK mailbox locked and ready
C: STAT
S: +OK 2 436
C: LIST
S: +OK mailbox has 2 messages
S: 1 232
S: 2 204
S: .
C: QUIT
S: +OK bye
S: (schliesst die Verbindung)
```

Listing 2: Abstrakte Klasse Base

```
1  import java.io.IOException;
2
3  public abstract class Base {
4
5          final String NL = "\r\n";
6          final String OK = "+OK ";
7          final String ERR = "-ERR ";
8
9          /* Wartet auf eingehende Verbindung vom POP3-Client */
10         void acceptConnection() throws IOException {...}
11
12         /* Trennt die Verbindung zum POP3-Client */
13         void disconnect() throws IOException {...}
14
15         /* Wartet bis ein Befehl vom verbundenen POP3-Client
16          * emfangen wurde und liefert die emfangene Befehlzeile
17          * als String zurueck */
18         String receive() throws IOException {...}
19
20         /* Sendet eine Antwortnachricht an den POP3-Client. */
21         void send(String answer) throws IOException {...}
22
23         /* Authentifizierung beim POP3-Server.
24          * Liefert bei Erfolg true, sonst false zurueck. */
25         boolean authentificate(String user,String password){...}
26
27         /* Liefert die Anzahl und die Gesamtgroesse der E-Mails
28          * in der Mailbox des aktuell autorisierten Benutzers.
29          * Beispiel: "2 436"    */
30         String stat() throws POP3Exception {...}
31
32         /* Liefert eine Liste, bestehend aus der ID und
33          * der Groesse (in Bytes) jeder Nachricht im Postfach.
34          * Beispiel: "1 232", "2 204"   */
35         String[] list() throws POP3Exception {...}
36
37         /* Muss in POP3Server-Klasse implementiert werden */
38         abstract public void run() throws IOException;
39 }
```

```
1  public class POP3Server extends Base{
2     public void run() throws IOException{
3        while(true){
4           acceptConnection();
5           send(OK + "example.com POP3-Server" + NL);
6           string user = receive();
7           send(OK + "Please enter password" + NL);
8           string pass = receive();
9           if(authentificate(user, pass)){
10             send(OK + "mailbox locked and ready" + NL);
11          }else{
12             send(ERR + "wrong password or user" +NL);
13             continue;
14          }
15          boolean ready = true;
16          while(ready){
17          string order = receive();
```

```
18          if(order.equals("STAT")){
19              try{
20                  String sta = stat();
21                  send(OK + sta + NL);
22              }catch{
23                  throws new POP3Exception();
24              }
25          } else if (order.equals("LIST")){
26              try{
27                  String mails[] = list();
28                  send(OK + "mailbox has " + mails.length + " messages\n");
29                  for(int i = 0; i < mails.length; i++){
30                      send(mails[i] + "\n");
31                  }
32                  send(NL);
33              }catch{
34                  throws new POP3Exception();
35              }
36          } else if (order.equals("QUIT")){
37              send(OK + "bye" + NL);
38              ready = false;
39          }
40      }
41  }
42 }
```

2 Socket-Programmierung (30 Punkte)

Programmieren Sie einen vereinfachten POP3-Client in JAVA, der eine Verbindung zu einem POP3-Server aufbauen, sich authentifizieren und die Anzahl der vorhandenen E-Mails überprüfen kann. Leiten Sie hierzu die Klasse `Pop3` von `Base` ab und implementieren Sie die abstrakten Methoden. Halten Sie sich an das in Abbildung 1 gezeigte Verhalten. Gehen Sie davon aus, dass die Methoden `init()` und `test()` von außen aufgerufen werden, d.h., Sie benötigen kein `main()`. Nehmen Sie außerdem an, dass `BufferdReader in` und `BufferdWriter out` bei `authentificate();` und `countMails();` geöffnet sind. Exceptions müssen berücksichtigt, aber nicht behandelt werden. Berücksichtigen Sie die Konstanten, Variablen, Methoden und Kommentare von `Base`:

```
abstract class Base {
   abstract void init() throws Exception; // siehe Statechart!
   abstract void test() throws Exception; // siehe Statechart!

   // Authentifizierung beim POP3-Server. erfolgreich --> true, sonst false
   abstract boolean authentificate() throws Exception;
   // Frägt die Anzahl der E-Mails beim POP3-Server ab
   abstract int countMails() throws Exception;

   // Verwenden Sie diese Konstanten in den dahinter stehenden Methoden;
   final static String POP3_SERVER = "192.168.5.122";// für init();
   final static String USER        = "user";         // für authentificate();
   final static String PASSWORD    = "geheim";       // für authentificate();

   // Verwenden Sie diese Variablen für die Kommunikation mit dem POP3-Server
   Socket socket;          // für init();
   BufferedReader in;      // für init(); authentificate(); countMails();
   BufferedWriter out;     // für init(); authentificate(); countMails();

   // Verwenden Sie diese Variablen und Methoden wie im Statechart
   boolean authOK = false;      // Authentifizierung erfolgreich? ja --> true
   void print(int num){}        // Ausgabe einer Zahl
   void end() {}                // Beenden des POP3-Clients
}
```

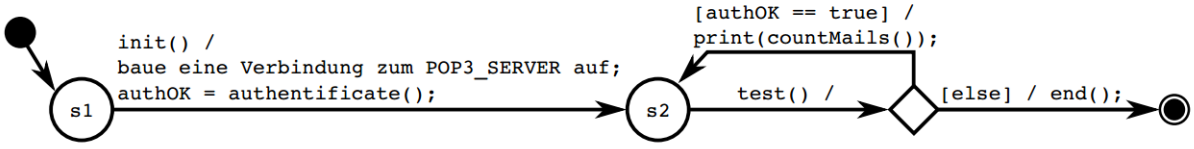


Abbildung 1: Vollständiges Statechart eines vereinfachten POP3-Clients

```
class Pop3 extends Base{

    void init() throws Exception{
        socket = new Socket(POP3_SERVER, 110);
        out = new DataOutputStream(socket.getOutputStream());
        in = new BufferedReader(new InputStreamReader(socket.getInputStream()
    ));
        authOK = authentificate();
    }

    boolean authentificate() throws Exception{
        out.writeBytes(USER);
        string s = in.readLine();
        if(!s.beginsWIth("+OK"))
            return false;
        out.writeBytes(PASSWORD);
        s = in.readLine();
        if(!s.beginsWIth("+OK"))
            return false;
    return true;
    }

    int countMails() throws Exception{
        out.writeBytes("list");
        int counter = 0;
        while(!in.readline().equals(".")){
            in.readLine();
            counter++;
        }
    return counter;
    }

    void test() throws Exception;{
        if(authOK == true){
            print(countMails());
        }else{
            end();
        }
    }
}
```

6 TCP Überlastkontrolle

6.1 Erkennen

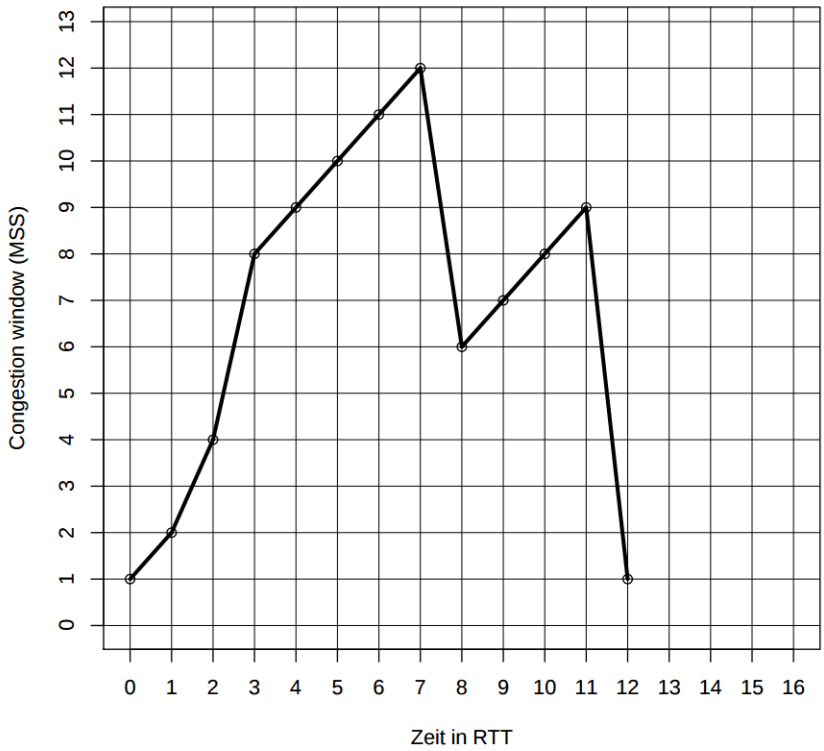


Abbildung 1: TCP-Fenstergröße als eine Funktion der Zeit

2 Transportschicht (26 Punkte gesamt)

2.1 TCP-Überlastkontrolle (7×2 Punkte)

Abbildung 1 stellt den Verlauf des *Congestion Windows* über die Zeit in *Round Trip Times* (RTT) dar. Nehmen Sie an, dass sich im Zustand des *Additive Increase* das *Congestion Window* um exakt eine *Maximum Segment Size* (MSS) pro RTT erhöht. Weiterhin hat der Threshold einen endlichen Initialwert.

Identifizieren Sie die Zeitintervalle, in denen TCP Slow Start in Betrieb ist.
$0RTT - 3RTT$ (zu erkennen am exponentiellen Anstieg)
In Abbildung 1 ist nach der t = 7 RTT ein Fehler aufgetreten. Wie wurde dieser erkannt? Begründen Sie Ihre Antwort.
3 doppelte Acks wurden gesendet (d.h. Verlust gemerkt) Erkennt man daran, dass: CW halbiert wurde (Ausserdem wird hier der Threshold gesetzt).
In Abbildung 1 ist nach der t = 11 RTT ein Fehler aufgetreten. Wie wurde dieser erkannt? Begründen Sie Ihre Antwort.

es ist ein Timeout passiert. => congestian window = MSS =1
Zeit für zurückkommende Acks ist abgelaufen
Welchen Wert hat der Threshold zum Zeitpunkt t = 2 RTT? Begründen Sie ihre Antwort kurz.
Der Threshold hat zum Zeitpunkt t=2RTT d.h den Wert 8. Er wurde mit einem endlichen Wert initialisiert. Erkennbar daran, dass ab 3RTT von Slow Start in linear growth umgeschalten wird (Wenn Threshold erreicht, wir umgeschalten)
Welchen Wert hat der Threshold zum Zeitpunkt t = 9 RTT? Begründen Sie ihre Antwort kurz.
Threshold wird nach einem Verlust auf CW/2 gesetzt (zum Zeitpunkt des Verlusts 12 => 6)
Wurde vor der Zeit t = 12 RTT das 36. Segment mit Daten bereits verschickt? Falls ja, zu welcher RTT-Zeit?
RTT = 6 (einfach die MSS bis dahin zusammenaddieren)
Nehmen Sie nun an, dass keine weiteren Fehler auftreten. Zeichnen Sie das Kurvenverhaltenfur den Zeitraum t (12, 16] RTT in Abbildung 1 ein.
=> 2,4,5,6 (treshold=5, mit treshold=4(abgerundet))

6.2 Zeichnen

2.3 TCP Überlastkontrolle (8 Punkte)

Zeichnen Sie in Abbildung 3 den Verlauf des Congestion Windows über die Zeit in Round Trip Times (RTT) möglichst exakt ein. Sie können vereinfachend annehmen, dass sich im Zustand des *Additive Increase* das Congestion Window um exakt eine MSS pro RTT erhöht. Beschreiben Sie für die drei Zeiträume $0-4$, $5-13$ und $14-25$ das Kurvenverhalten und nennen Sie die zugrunde liegende Vorschrift. Geben Sie für jeden Zeitraum den Threshold an.

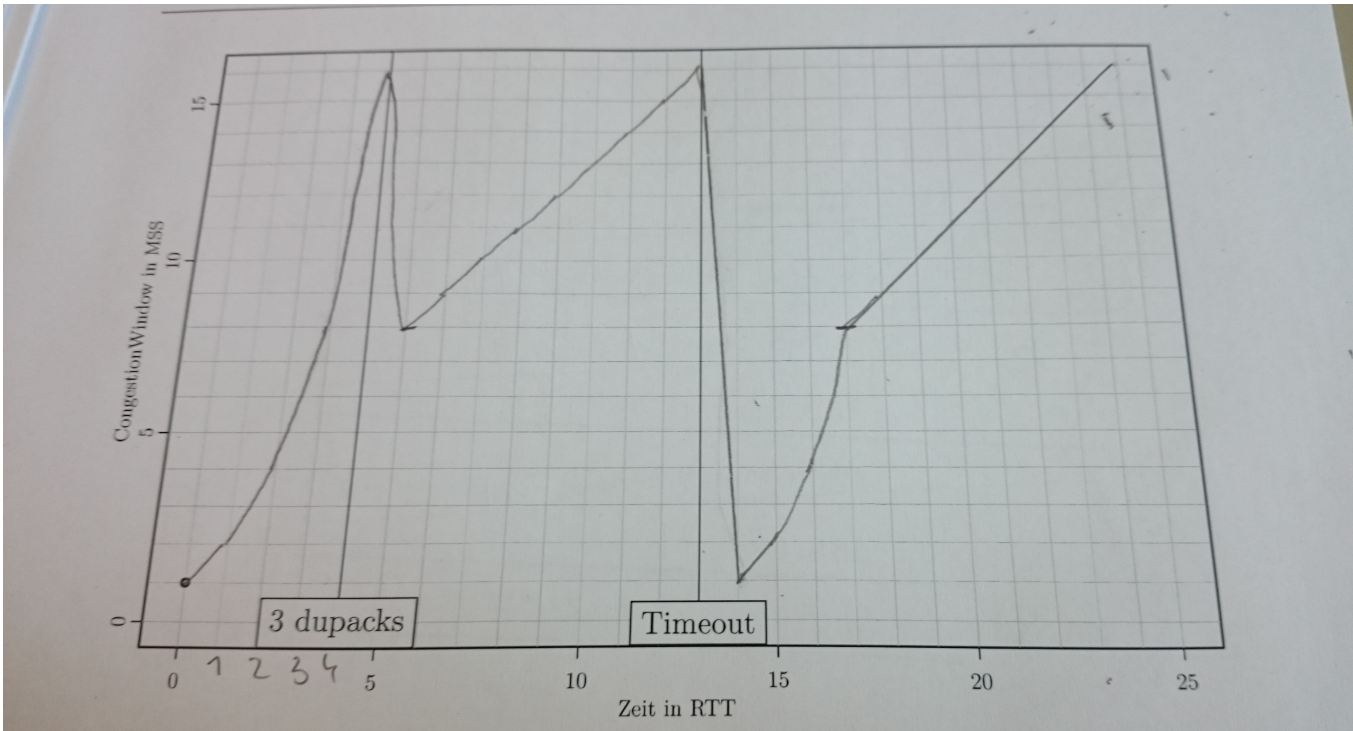


Abbildung 3: Zeitlicher Ablauf des CongestionWindow

Inhaltsverzeichnis

1 RK - Zusammenfassung

1.1 Beispiele für Leitungs-Codierungen

- NRZ-Code (unipolar/bipolar)
- RZ-Code (unipolar/bipolar)
- Manchesterkodierung
- Blockcodes
- Digitale Frequenzmodulation
- MLT-3-Code (gleichstromarme Umsetzung von Binärfolgen auf drei Spannungspegel)
- AMI-Code
- Digitale Frequenzmodulation

1.2 Allgemeine Aufgabe der Leitungscodierung

Die Aufgabe der Leitungscodierung (bzw. Leitungskodierung) ist, das zu übertragende Signal spektral zu formen, um es so möglichst optimal an die Eigenschaften eines Übertragungsmediums anzupassen. So kann beispielsweise der Gleichspannungsanteil unterdrückt werden. Daneben wird eine Taktrückgewinnung möglich.

1.3 RZ-Code

Beim Return-to-Zero-Code (kurz RZ-Code; Englisch für Rückkehr zur Null) RZ-Codierung einer binären Folge Der RZ-Code ist eine Weiterentwicklung des NRZ-Codes (Non-Return-to-Zero), bei dem die zwei Pegelwerte gleichbedeutend mit dem Dualwert der zu übertragenden Ziffer sind. Der Nachteil des NRZ-Code ist, dass es beim Übertragen einer längeren Serie von Nullen oder einer Serie von Einsen keine Pegeländerungen gibt. Dadurch ist es für den Empfänger in diesem Zeitraum nicht möglich, den Takt aus dem Signal zurückzugewinnen. Beim RZ-Code kehrt man beim Übertragen einer logischen 1 mit dem Pegel +1 nach dem halben Takt zum Pegel 0 zurück, bei Übertragung einer logischen 0 wird der Pegel -1 für eine halbe Periode übertragen und nachfolgend zum Pegel 0 zurückgekehrt. Dadurch gibt es beim Übertragen eines Bits garantiert eine Pegeländerung welche der Empfänger zur Taktrückgewinnung (Synchronisierung) nutzen kann. Nachteilig gegenüber dem NRZ-Code ist, dass eine doppelt so hohe Bandbreite benötigt wird.

1.3.1 Unipolare RZ-Codierung

Eine Sonderform stellt die unipolare RZ-Kodierung dar. Der Vorteil besteht darin, dass nur zwei Pegelwerte (+1 und 0) als Symbole benötigt werden und diese Codierung daher mit herkömmlichen Digitalschaltungen leicht realisiert werden kann. Der Nachteil besteht darin, dass bei der Übertragung einer langen logisch-0-Folge, welche mit konstantem Pegel 0 codiert wird, keine Signaländerung erfolgt und damit eine Synchronisierung seitens des Empfängers unmöglich ist.

1.3.2 Bipolare RZ-Codierung

Die bipolare RZ-Codierung ist eng mit der unipolaren RZ-Codierung verwandt, verwendet allerdings wie die RZ-Codierung drei Pegel: Der Zustand logisch-0 wird wie bei der unipolaren RZ-Codierung immer mit Pegel 0 übertragen. Der Zustand logisch-1 wird alternierend mit dem Pegel +1 und -1 übertragen.

1.4 Protokoll-Stapel

Ein Protokollstapel oder Protokollturm ist eine konzeptuelle Architektur von Kommunikationsprotokollen. Anschaulich sind die einzelnen Protokolle dabei als fortlaufend nummerierte Schichten (layers) eines Stapels (stacks) übereinander angeordnet. Jede Schicht benutzt dabei zur Erfüllung ihrer speziellen Aufgabe die jeweils tiefere Schicht im Protokollstapel. Jedes Netzwerkprotokoll entfernt beim Empfang aus den Daten diejenigen Steuerinformationen, die nur für dieses Protokoll selbst bestimmt sind, und übergibt die verbliebenen Daten dem nächsthöheren Netzwerkprotokoll. In der Senderichtung werden die Steuerinformationen hinzugefügt, bevor sie dem nächsttieferen Netzwerkprotokoll übergeben werden - eine Nachricht trägt also auf der Leitung sämtliche Header der darüberliegenden Schichten. Eine HTTP-Nachricht, die via Ethernet versandt wird, lässt sich wie folgt veranschaulichen:

- Ethernet-Frame
– IP-Paket
— TCP-Segment
—- HTTP-Nachricht

1.5 OSI-Modell

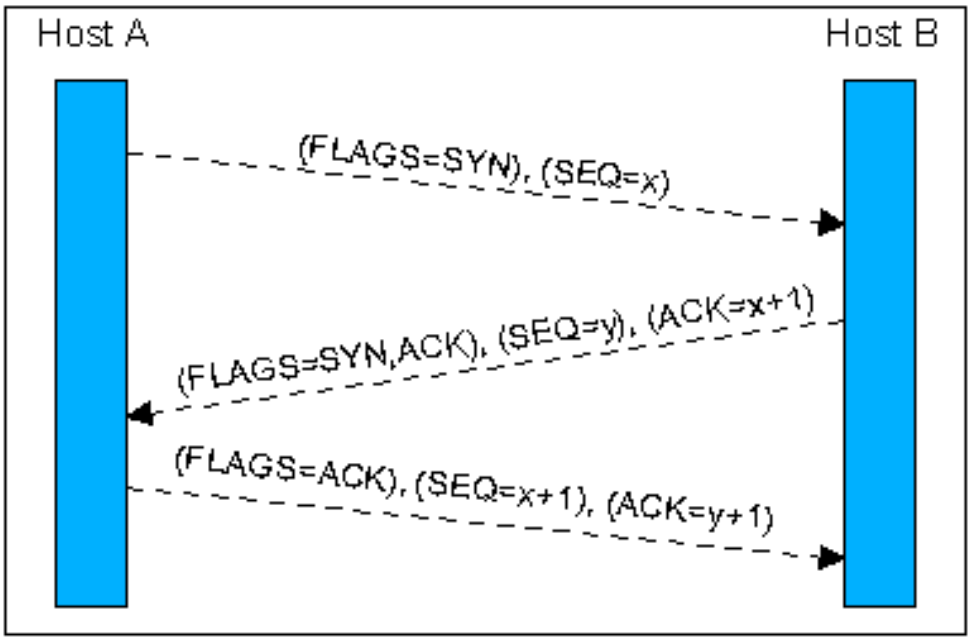
OSI-Schicht		Einordnung	DoD-Schicht	Einordnung	Protokollbeispiel	Einheiten	Kopplungselemente
7	Anwendungen (Application)	Anwendungs-orientiert	Anwendung	Ende zu Ende (Multihop)	HTTP FTP HTTPS SMTP LDAP NCP	Daten	Gateway, Content-Switch, Proxy, Layer-4-7-Switch
6	Darstellung (Presentation)						
5	Sitzung (Session)						
4	Transport (Transport)	Transport-orientiert	Transport		TCP UDP SCTP SPX	TCP = Segmente UDP = Datagramme	
3	Vermittlung (Network)		Internet	Punkt zu Punkt	ICMP IGMP IP IPsec IPX	Pakete	Router, Layer-3-Switch
2	Sicherung (Data Link)		Netzzugriff		Ethernet Token Ring FDDI MAC ARCNET	Rahmen (Frames)	Bridge, Switch
1	Bitübertragung (Physical)					Bits, Symbole, Pakete	Netzwerkkabel, Repeater, Hub

- 7 **Dienste, Anwendungen und Netzmanagement.** Die Anwendungsschicht stellt Funktionen für die Anwendungen zur Verfügung. Auf dieser Ebene findet auch die Dateneingabe und -ausgabe statt.
- 6 Die **Darstellungsschicht** setzt die systemabhängige Darstellung der Daten (zum Beispiel ASCII, EBCDIC) in eine unabhängige Form um. Auch Datenkompression + Verschlüsselung.
- 5 Die **Sitzungsschicht** sorgt für die Prozesskommunikation zwischen zwei Systemen. Hier findet sich unter anderem das Protokoll RPC (Remote Procedure Call). Um Zusammenbrüche der Sitzung und ähnliche Probleme zu beheben, stellt die Sitzungsschicht Dienste für einen organisierten und synchronisierten Datenaustausch zur Verfügung.
- 4 Zu den Aufgaben der **Transportschicht** zählen die Segmentierung des Datenstroms und die Stauvermeidung (engl. congestion avoidance).
- 3 Die **Vermittlungsschicht** sorgt bei leitungsorientierten Diensten für das Schalten von Verbindungen und bei paketorientierten Diensten für die Weitervermittlung von Datenpaketen. + schließt die Wegesuche (Routing) zwischen den Netzwerkknoten ein.
- 2 Aufgabe der **Sicherungsschicht** ist es, eine weitgehend fehlerfreie Übertragung zu gewährleisten und den Zugriff auf das Übertragungsmedium zu regeln. Dazu dient das Aufteilen des Bitdatenstromes in Blöcke – auch als Frames oder Rahmen bezeichnet – und das Hinzufügen von Prüfsummen im Rahmen der Kanalkodierung.

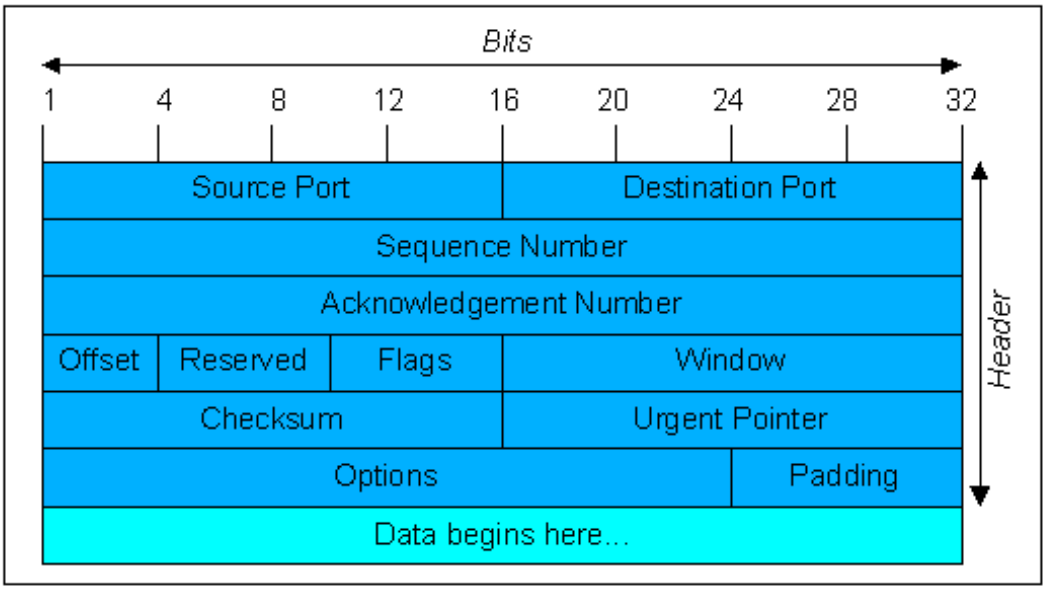
- 1 Die **Bitübertragungsschicht** ist die unterste Schicht. Diese Schicht stellt mechanische, elektrische und weitere funktionale Hilfsmittel zur Verfügung, um physische Verbindungen zu aktivieren bzw. zu deaktivieren, sie aufrechtzuerhalten und Bits darüber zu übertragen.

Motivation Wegen der Vielzahl von Problemen und Aufgaben hat man sich entschieden, diese in verschiedene Ebenen (Schichten) aufzuteilen. Beim OSI-Modell sind es sieben Schichten mit festgelegten Anforderungen. Auf jeder einzelnen Schicht setzt jeweils eine Instanz die Anforderungen um.

1.6 TCP-Header



Dreiwege-Handshake (hier Verbindungsaufbau).



Der TCP-Header.

- **Source-/Destination-Port** Die Felder Source Port (Quellport) und Destination Port (Zielport) adressieren die Endpunkte der Verbindung.
- **Sequence Number, Acknowledgement Number** Die Sequenznummer und die

Bestätigungsnummer sind jeweils 32-Bit-Zahlen. Die Sequenznummer gilt in Senderichtung, die Bestätigungsnummer für Empfangsquittungen. Jeder der beiden TCP-Verbindungspartner generiert beim Verbindungsaufbau eine Sequenznummer, die sich während des Zeitraums der Verbindung nicht wiederholen darf. Die Nummer gibt allerdings nicht an, welches Byte zuletzt korrekt empfangen wurde, sondern welches Byte als nächstes zu erwarten ist.

- **Offset** Das Feld Offset (oder auch Header Length) gibt die Länge des TCP-Headers in 32-Bit Worten an. Dies entspricht dem Anfang der Daten im TCP-Segment. Das Feld ist notwendig, da der Header durch das Optionsfeld eine variable Länge hat.
- Mit den sechs 1-Bit-Flags im Flags-Feld werden bestimmte Aktionen im TCP-Protokoll aktiviert:
  - **URG** Wird das Flag URG auf 1 gesetzt, so bedeutet dies, daß der Urgent Pointer (Dringendzeiger) verwendet wird.
  - **ACK** Das ACK-Bit wird gesetzt, um anzugeben, dass die Bestätigungsnummer im Feld Acknowledgement Number gültig ist. Ist das Bit auf 0 gesetzt, enthält das TCP-Segment keine Bestätigung, das Feld Acknowledgement Number wird ignoriert.
  - **PSH** Ist das PSH-Bit gesetzt, so werden die Daten in dem entsprechenden Segment sofort bei Ankunft der adressierten Anwendung bereitgestellt ohne sie zu puffern.
  - **RST** Das RST-Bit dient dazu eine Verbindung zurückzusetzen, falls ein Fehler bei Übertragung aufgetreten ist. Dies kann sowohl der Fall sein, wenn ein ungültiges Segment übertragen wurde, ein Host abgestürzt ist oder der Versuch eines Verbindungsaufbaus abgewiesen werden soll.
  - **SYN** Das SYN-Flag (Synchronize Sequenze Numbers) wird verwendet, um Verbindungen aufzubauen. Zusammen mit der Acknowledgement Number und dem ACK-Bit wird die Verbindung im Form eines Dreiwege-Handshake aufgebaut (siehe oben).
  - **FIN** Das FIN-Bit dient zum Beenden einer Verbindung. Ist das Bit gesetzt, gibt dies an, daß der Sender keine weiteren Daten zu Übertragen hat. Das Segment mit gesetztem FIN-Bit muß quittiert werden.
- **Window** Das Feld Fenstergröße enthält die Anzahl Bytes, die der Empfänger ab dem bereits bestätigten Byte empfangen kann. Mit der Angabe der Fenstergröße erfolgt in TCP die Flußsteuerung. Das TCP-Protokoll arbeitet nach dem Prinzip eines Schiebefensters mit variabler Größe (Sliding Window). Jede Seite einer Verbindung darf die Anzahl Bytes senden, die im Feld für die Fenstergröße angegeben ist, ohne auf eine Quittung von der Empfängerseite zu warten.
- **Checksum** Die Prüfsumme prüft den Protokollkopf, die Daten und den Pseudo-Header. Dieser Pseudo-Header besteht aus der Ziel-IP, der Quell-IP, der TCP-Protokollkennung (0x0006) und der Länge des TCP-Headers inkl. Nutzdaten (in

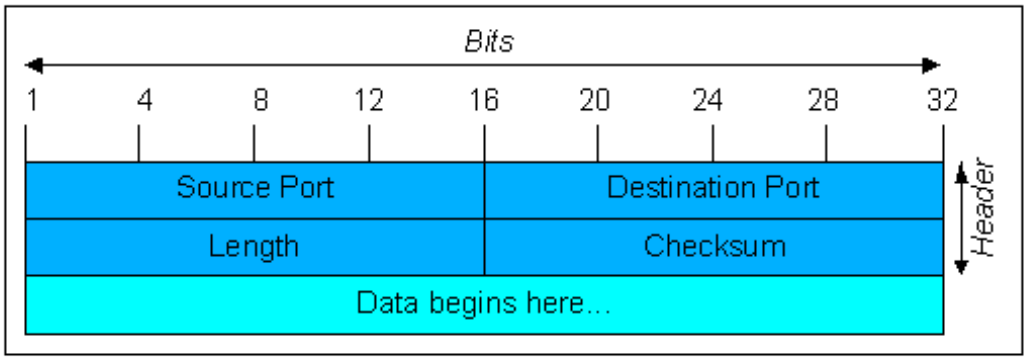
Bytes). Der Algorithmus für die Bildung der Prüfsumme ist einfach: alle 16-Bit Wörter werden im 1er-Komplement addiert und die Summe ermittelt. Bei der Berechnung ist das Feld Checksum auf Null gesetzt und das Datenfeld wird bei ungerader Länge um ein Nullbyte aufgefüllt. Führt der Empfänger des Segments die Berechnung auf das gesamte Segment aus - inklusive des Feldes für die Prüfsumme - sollte das Ergebnis 0 sein. Der Pseudo-Header enthält die 32-bit großen IP-Adressen der Quell- und Zielmaschine sowie die Protokollnummer (für TCP 6) und die Länge des TCP-Segments. Die Einbeziehung der Felder des Pseudo-Headers in die Prüfsummenberechnung hilft, durch IP falsch zugeteilte Pakete zu erkennen. Die Verwendung von IP-Adressen auf der Transportebene stellt allerdings eine Verletzung der Protokollhierarchie dar.

- **Urgent Pointer** Der Urgent-Zeiger ergibt zusammen mit der Sequenznummer einen Zeiger auf ein Datenbyte. Dies entspricht einem Byteversatz zu einer Stelle, an der dringende Daten vorgefunden werden. TCP signalisiert damit, daß sich an einer bestimmten Stelle im Datenstrom wichtige Daten befinden, die sofort gelesen werden sollten. Das Feld wird nur gelesen, wenn auch das Urgent-Flag (s.o.) gesetzt ist.

- **Options** Das Options-Feld soll eine Möglichkeit bieten Funktionen bereitzustellen, die im normalen TCP-Protokollkopf nicht vorgesehen sind. In TCP sind drei Optionen definiert: End of Option List, No-Operation und Maximum Segment Size. Die wichtigste dieser drei Optionen ist die Maximale Segmentgröße. Mit dieser Option kann ein Host die maximale Anzahl Nutzdaten übermitteln, die er annehmen will bzw. annehmen kann

- **Padding** Das Feld Padding wird verwendet, um sicherzustellen, daß der Header an einer 32-Bit Grenze endet und die Daten an einer 32-Bit Grenze beginnen. Das Füllfeld wird mit Nullen gefüllt.

1.7 UDP



Der UDP-Header.

1.8 QoS

Quality of Service (QoS) oder Dienstgüte beschreibt die Güte eines Kommunikationsdienstes aus der Sicht der Anwender. Standard IEEE 802.1p Anforderungen:

- Ein Anwender möchte zuverlässig mit dem gewünschten Ziel verbunden werden und nach Ende der Kommunikation zuverlässig getrennt werden.
- Der Verbindungsaufbau soll rasch erfolgen.
- Probleme beim Verbindungsaufbau (z. B. Ziel-Teilnehmer nicht erreichbar) sollen dem Anwender schnellstmöglich mitgeteilt werden.
- Eine Kommunikationsverbindung soll stabil bestehen bleiben.
- Die Kommunikationsteilnehmer wollen sich verstehen können.
- Die Informationen sollen vollständig und ohne Fehler übertragen werden.
- Es sollen keine Informationen anderer Kommunikationsteilnehmer und keine Störungen übertragen werden.
- Die Kommunikation soll möglichst originalgetreu vor sich gehen.
- Es sollen keine langen Wartezeiten während der Kommunikation bestehen.
- Die Abrechnung der Kommunikation soll dem korrekten Zeit- und Datenumfang entsprechen.

1.9 Peer-to-Peer

In einem reinen Peer-to-Peer-Netz sind alle Computer gleichberechtigt und können sowohl Dienste in Anspruch nehmen, als auch zur Verfügung stellen.
Typische, aber nicht notwendige Charakteristika von Peer-to-Peer-Systemen sind:

- Peers weisen eine hohe Heterogenität bezüglich der Bandbreite, Rechenkraft, Online-Zeit, ... auf.
- Die Verfügbarkeit und Verbindungsqualität der Peers kann nicht vorausgesetzt werden (”Churn”).
- Peers bieten Dienste und Ressourcen an und nehmen Dienste anderer Peers in Anspruch (Client-Server-Funktionalität).
- Dienste und Ressourcen können zwischen allen teilnehmenden Peers ausgetauscht werden.
- Peers bilden ein Overlay-Netzwerk und stellen damit zusätzliche Such-Funktionen zur Verfügung.

- Peers haben eine signifikante Autonomie (über die Ressourcenbereitstellung).
- Das P2P-System ist selbstorganisierend.
- Alle übrigen Systeme bleiben konstant intakt und nicht skaliert.

P2P-Systeme lassen sich in unstrukturierte und strukturierte P2P-Systeme unterteilen. Z.B. Filesharing

1.10 Überlastkontrolle

Überschreitet die Anzahl der Pakete, die dem Netz insgesamt aufgebürdet werden, einen gewissen Punkt, so wird das Netz überlastet und seine Leistung sinkt drastisch, weil es zuviel Zeit damit verbringen muss, Konflikte und Überlastsituationen zu behandeln. Maßnahmen gegen dieses Phänomen bezeichnen wir als Überlastkontrolle

1.11 Flusskontrolle

Überschreitet die Anzahl der Pakete, die ein Sender auf einer logischen Verbindung dem Netz bei insgesamt niedriger Auslastung übergibt, die Aufnahmekapazität des Empfängers, so wird dieser überlastet bzw. Pakete können nicht ordnungsgemäß ausgeliefert werden. Maßnahmen gegen dieses Phänomen bezeichnen wir als Flusskontrolle Flusskontrolle kann man einfach realisieren: In einer Wechselbeziehung vereinbaren die Stationen, die in aller Regel sowohl Sender als auch Empfänger sind, einen Flusskontrollparameter (Credit), der die Anzahl der Pakete angibt, die ein Sender senden darf, ohne von einem Empfänger eine Empfangsbestätigung erhalten zu haben. Alle Übertragungsprotokolle arbeiten mit Flusskontrolle, allerdings mit unterschiedlichen Algorithmen.

1.12 ICMP

ICMP benutzt IP als Kommunikationsbasis, indem es sich selbst als Protokoll einer höheren Schicht interpretiert, d. h. ICMP-Nachrichten werden in IP-Paketen gekapselt. ICMP erkennt einige Fehlerzustände, macht aber IP zu keinem zuverlässigen Protokoll.

1.13 SMTP

Das **Simple Mail Transfer Protocol** ist ein Protokoll der Internetprotokollfamilie, das zum Austausch von E-Mails in Computernetzen dient. Bentutzt in Normalfall TCP.

Client	Server	Erläuterung
`telnet mail.example.com 25`		Client ruft Server
	`220 service ready`	Server meldet sich bereit
`HELO foobar.example.net`		Client nennt seinen Namen
	`250 OK`	Server bestätigt
`MAIL FROM:<sender@example.org>`		Client nennt Absenderadresse
	`250 OK`	Server bestätigt
`RCPT TO:<receiver@example.com>`		Client nennt Empfängeradresse
	`250 OK`	Server bestätigt
`DATA`		Client kündigt Inhalt der E-Mail an
	`354 start mail input`	Server bereit für diesen längeren Vorgang

`From: <sender@example.org>`
`To: <receiver@example.com>`
`Subject: Testmail`
`Date: Thu, 26 Oct 2006 13:10:50 +0200`
 	
`Lorem ipsum dolor sit amet,`
`consectetur adipisici elit, sed`
`eiusmod tempor incidunt ut labore et`
`dolore magna aliqua.`
`.`
		Client sendet Inhalt der E-Mail und markiert das Ende durch eine Zeile, die nur einen Punkt enthält.
		(Zwischen Header und Textkörper muss eine Leerzeile vorhanden sein, sonst wird beim Empfänger kein Textkörper angezeigt.)

	`250 OK`	Server bestätigt und übernimmt die Verantwortung für die Nachricht
`QUIT`		Client fordert Verbindungstrennung an

1.14 POP3

Das Post Office Protocol (POP) ist ein Übertragungsprotokoll, über das ein Client E-Mails von einem E-Mail-Server abholen kann. Bentutzt in Normalfall TCP.

Client	Server
	(wartet auf Verbindungen auf TCP Port 110)
(öffnet Verbindung)	
	+OK example.com POP3-Server
USER wiki@example.com	
	+OK Please enter password
PASS passwort_im_klartext	
	+OK mailbox locked and ready
STAT	
	+OK 1 236
LIST	
	+OK mailbox has 1 messages (236 octets) 1 236 .
RETR 1	
	+OK message follows Date: Mon, 18 Oct 2004 04:11:45 +0200 From: Someone <someone@example.com> To: wiki@example.com Subject: Test-E-Mail Content-Type: text/plain; charset=us-ascii; format=flowed Content-Transfer-Encoding: 7bit Dies ist eine Test-E-Mail .
DELE 1	
	+OK message marked for delete

Vorteile

- Es ist keine ständige Verbindung zum Mailserver notwendig
- Die Verbindung wird bei Bedarf vom Client aufgebaut und beendet
- Nach der Anmeldung werden alle E-Mails vom Mailserver heruntergeladen Nachteile:
- Eine Synchronisierung zwischen den Clients findet nicht statt. Wird eine Mail gelöscht oder als ”Gelesen” markiert, wird diese Information nicht auf andere Mail-Clients übertragen.

1.15 Ethernet-Packet (1 octet = 8bit = 1Byte)

Aufbau eines Ethernet-Pakets mit maximalen IPv4- / TCP-Daten

Schicht 4: TCP-Segment								TCP-Header	*Nutzlast (1460 bytes)*		
Schicht 3: IP-Paket							IP-Header	*Nutzlast (1480 bytes)*			
Schicht 2: Ethernet-Frame			MAC-Empfänger	MAC-Absender	802.1Q-Tag (opt.)	EtherType	*Nutzlast (1500 bytes)*			Frame Check Sequence	
Schicht 1: Ethernet-Paket+IPG	Präambel	Start of Frame	*Nutzlast (1518/1522 bytes)*								Interpacket Gap
Oktette	7	1	6	6	(4)	2	20	20	6–1460	4	12

802.3 Ethernet packet and frame structure

Layer	Preamble	Start of frame delimiter	MAC destination	MAC source	802.1Q tag (optional)	Ethertype (Ethernet II) or length (IEEE 802.3)	Payload	Frame check sequence (32-bit CRC)	Interpacket gap
	7 octets	1 octet	6 octets	6 octets	(4 octets)	2 octets	46(42)[b]–1500 octets	4 octets	12 octets
Layer 2 Ethernet frame			← 64–1518(1522) octets →						
Layer 1 Ethernet packet	← 72–1526(1530) octets →								

- The **preamble** of an Ethernet packet consists of a 56-bit (seven-byte) pattern of alternating 1 and 0 bits, allowing devices on the network to easily synchronize their receiver clocks, which is followed by the SFD to mark a new incoming frame.
- The **SFD** is the eight-bit (one-byte) value marking the end of the preamble (=10101011)
- The **minimum payload** is 42 octets when an 802.1Q tag is present and 46 octets when absent. The maximum payload is 1500 octets.
- The **frame check sequence (FCS)** is a four-octet cyclic redundancy check (CRC) that allows detection of corrupted data within the entire frame as received on the receiver side.
- **Interpacket gap** is idle time between packets.

1.16 RTT

Die Round Trip Time (RTT, englisch für Rundreisezeit) bzw. Paketumlaufzeit gibt die Zeit an, die ein Datenpaket (Datagramm) in einem Rechnernetz benötigt, um von der Quelle zum Ziel und zurück zu reisen. Es handelt sich also um die Summe aus Laufzeit von Punkt A nach Punkt B und der Laufzeit von Punkt B nach Punkt A.

1.17 Routingverfahren

1.17.1 Distanzvektorprotokoll

Das prinzipielle Vorgehen eines Distanzvektorprotokolls

- Erzeuge eine Kostenmatrix, welche Router über welche Nachbarn und zu welchen Kosten erreichbar sind und anfangs nur die (bekannten) Kosten zu direkten Nachbarn enthält.
- Erzeuge eine Aufstellung mit Informationen, welche Router wir zu welchen Kosten am besten erreichen können und schicke sie an alle Nachbarn.
- Warte auf Aufstellungen dieser Art von anderen Routern, rechne diese dann in die eigene Kostenmatrix ein.
- Ändern sich dadurch die minimalen Kosten, zu denen wir einen Router erreichen können: fahre mit Schritt 2 fort, sonst mit Schritt 3.

1.17.2 Dijkstra/Forward-Search

Routingverfahren - Aufgabe 4

Abbildung 3 zeigt ein Netzwerk mit fünf Knoten. Simulieren Sie das Dijkstra-Verfahren in Tabelle 2 und den Forward-Search-Algorithmus in Tabelle 3, um minimale Pfade zwischen den Knoten zu erhalten. Verwenden Sie für das Dijkstra-Verfahren Knoten D und für den Forward-Search-Algorithmus den Knoten B als Startknoten. Vervollständigen Sie die Forwarding-Tabelle 4 auf Basis von Tabelle 3 und dem Forward-Search-Algorithmus.
Bitte benutzen Sie dieselben Symbole wie in der Vorlesung.

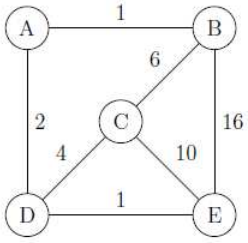


Abbildung 3: Netzwerk

Schritt	V'	D(A), p(A)	D(B), p(B)	D(C), p(C)	D(E), p(E)
0	D	2, A	∞, -	4, C	1, E
1	D, E	2, A	17, E	4, C	"
2	D, E, A	"	3, A	4, C	"
3	D, E, A, B	"	"	4, C	"
4	D, E, A, B, C	"	"	"	"
5					
6					

Tabelle 2: Dijkstra-Verfahren für Knoten D

Schritt	Bestätigte Liste	Vorläufige Liste
0	(B,0,-)	
1	(B,0,-)	(A,1,A), (C,6,C), (E,16,E)
2	(B,0,-), (A,1,A)	(C,6,C), (D,3,A), (E,16,E)
3	(B,0,-), (A,1,A), (D,3,A)	(C,6,C), (E,4,A)
4	(B,0,-), (A,1,A), (D,3,A), (E,4,A)	(C,6,C)
5	(B,0,-), (A,1,A), (D,3,A), (E,4,A), (C,6,C)	
6		

Tabelle 3: Forward-Search-Algorithmus für Knoten B

Ziel	Link
A	A
C	C
D	A
E	A

Tabelle 4: Forwarding-Tabelle für Knoten B

Zeichnen Sie auf Basis von Tabelle 2 (Dijkstra-Verfahren mit Startknoten D) die daraus resultierende minimale Baumstruktur in Abbildung 4.

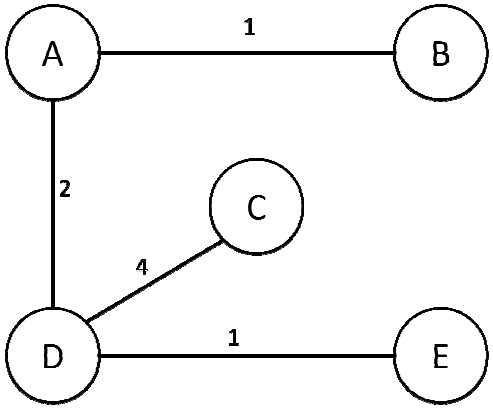
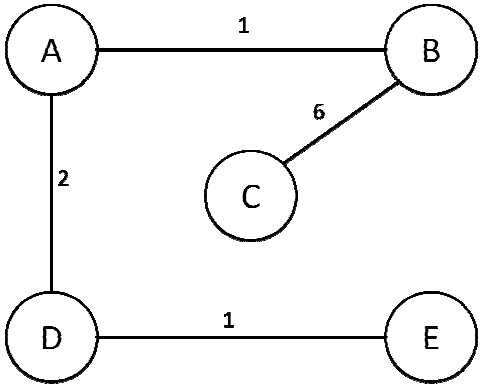


Abbildung 4: Minimal aufgespannter Baum

Wurde in beiden Teilaufgaben (Dijkstra-Verfahren und Forward-Search-Algorithmus) der gleiche Baum aufgespannt? Begründen Sie Ihre Antwort.



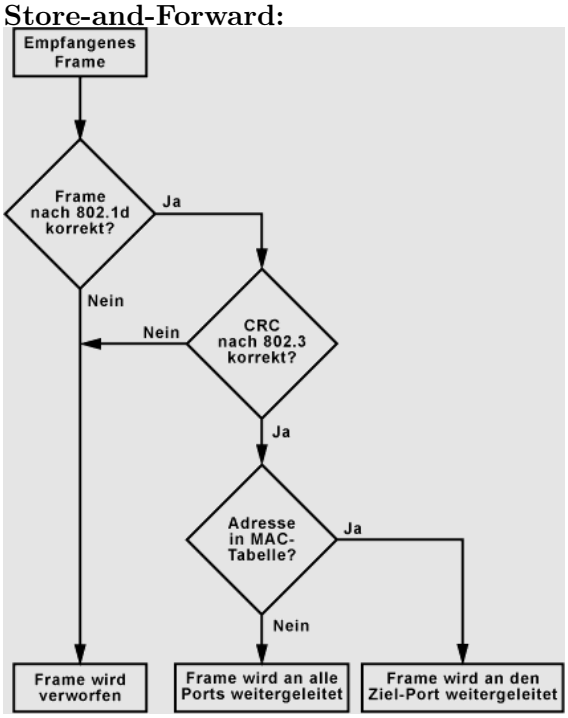
Nein, da unterschiedliche Kanten gewählt wurden.

1.18 Switching

Switching ist ein Mechanismus in paketorientierten Netzwerken, um für eingehende Datenpakete den richtigen Ausgang zu ermitteln. Typen:

- **Cut-Through:** Der Switch analysiert die Ethernet-Frames, bevor sie vollständig eingetroffen sind. Hat er die Ziel-Adresse identifiziert, wird das Frame sofort an den Ziel-Port ausgegeben. Die Latenz, die Verzögerungszeit zwischen Empfangen und Weiterleiten eines Frames, ist äußerst gering. Das Cut-Through-Verfahren verzichtet auf die vollständige Analyse der Frames.

- **Store-and-Forward:**



- **Adaptive-Cut-Through:** Je nach Implementierung gibt es Unterschiede bei diesem Switching-Verfahren. In jedem Fall wird auf eine Kombination aus Cut-Through und Store-and-Forward gesetzt. Im einen Fall werden die Frames mit Cut-Through weitergeleitet, aber anhand der Prüfsumme (CRC) geprüft. Wird eine bestimmte Fehlerrate überschritten wird automatisch auf Store-and-Forward umgeschaltet. Geht die Fehlerrate zurück, wird auf Cut-Through zurückgeschaltet.

- **FragmentFree-Cut-Through:** Dieses Verfahren stammt von Cisco und geht von einem Erfahrungswert bei fehlerhaften Frames aus. Man hat festgestellt, dass Übertragungsfehler am häufigsten innerhalb der ersten 64 Byte eines Frames auftreten. Deshalb überprüft ein, mit FragmentFree-Cut-Through arbeitender, Switch die ersten 64 Byte auf Fehler. Ist es fehlerfrei wird das Frame per Cut-Through weiterverarbeitet. Ist ein Fehler vorhanden, dann wird das Frame verworfen.

1.19 Go-Back-N

Der Sender kann dabei mehrere Dateneinheiten senden, ohne auf eine Quittung warten zu müssen. Wie viele das sind, hängt von der sogenannten Fenstergröße ab. Beträgt diese n, kann der Sender noch n-1 weitere Dateneinheiten absenden, bevor die Bestätigung für die erste Einheit durch den Empfänger erfolgt sein muss. Es können vom Empfänger auch mehrere Dateneinheiten auf einmal (kumulativ) bestätigt werden, so zeigt eine Quittung für n+i an, dass alle Einheiten von n bis n+i korrekt empfangen wurden. Kommt es beim Warten auf die Bestätigungen zu einem Timeout, so übermittelt der Sender alle Dateneinheiten in dem Fenster neu. Er geht also zurück zur letzten unbestätigten Sequenznummer N. Da es der Fall sein kann, dass lediglich eine Dateneinheit nicht ordnungsgemäß übertragen wurde und dennoch auch alle danach gesendeten erneut übertragen werden, wird an dieser Stelle Übertragungskapazität verschwendet.

The receiver process keeps track of the sequence number of the next frame it expects to receive, and sends that number with every ACK it sends. The receiver will discard any frame that does not have the exact sequence number it expects (either a duplicate frame it already acknowledged, or an out-of-order frame it expects to receive later) and will resend an ACK for the last correct in-order frame. Once the sender has sent all of the frames in its window, it will detect that all of the frames since the first lost frame are outstanding, and will go back to the sequence number of the last ACK it received from the receiver process and fill its window starting with that frame and continue the process over again. Go-Back-N ARQ is a more efficient use of a connection than Stop-and-wait ARQ, since unlike waiting for an acknowledgement for each packet, the connection is still being utilized as packets are being sent. In other words, during the time that would otherwise be spent waiting, more packets are being sent.

1.20 Selective-Repeat

Die andere allgemeine Fehlerbehandlungsstrategie bei ARQ-Protokollen ist Selektive Wiederholung. Hier wird ein fehlerhafter Rahmen verworfen, aber die danach erhaltenen Rahmen werden im Empfänger in einem Puffer abgelegt und bestätigt. Wenn beim Sender die Zeit abgelaufen ist, wird nur der älteste nicht bestätigte Rahmen erneut übertragen. Wenn dieser Rahmen korrekt ankommt, kann der Empfänger in der Folge alle im Puffer gespeicherten Rahmen an die Vermittlungsschicht übergeben.

1.21 Schiebefensterverfahren

Das Schiebefensterverfahren verfolgt das Ziel, die Kapazitäten der Leitung und des Empfängers optimal auszulasten, das heißt so viele Datenpakete (Datenframes) wie möglich zu senden. Dabei stellt das Verzögerung-Bandbreite-Produkt die maximale in der Übertragung befindliche Datenmenge dar, die gesendet werden kann ohne auf die erste Bestätigung zu warten.
Beim Schiebefensterverfahren führt der Sender permanent eine Liste von aufeinanderfolgenden Sequenznummern, die der Anzahl der Frames, die er senden darf, entspricht.

Sobald ein Datenpaket dem Empfänger erfolgreich zugestellt wird, sendet dieser dafür eine Bestätigung, auch als ACK-Signal bezeichnet, zurück, die den Sender dazu veranlasst, ein weiteres Frame zu übertragen. Falls der Sender innerhalb des Timeouts jedoch kein ACK erhält, versucht er das Frame erneut zu übertragen. Unter der Voraussetzung, dass das Verzögerungs-Bandbreiten-Produkt bereits erreicht ist, kann dann aber kein neues Frame übertragen werden, d. h. es kommt zu einem Stau in der Pipe. Das Sendefenster verschiebt sich mit jeder eingehenden Bestätigung, indem das bestätigte Frame aus dem Fenster herausfällt und ein neu zu sendendes Frame in das Fenster aufgenommen wird. Dadurch enthält das Fenster immer nur unbestätigte Frames. Für den Fall, dass Frames während der Übertragung verloren gehen, muss der Sender alle Datenpakete in seinem Speicher halten, um sie erneut übertragen zu können.

1.22 CRC-Check

Es folgt ein Beispiel, in dem für einen Binärcode von 5 Bit der CRC berechnet und überprüft werden soll. Das Generatorpolynom (CRC-Polynom) lautet 110101 ($1x^5 + 1x^4 + 0x^3 + 1x^2 + 0x^1 + 1x^0$) und ist somit 5. Grades. Der zu übertragenden Bitfolge, welche auch als Rahmen (engl. frame) bezeichnet wird, werden n Nullen angehängt (Rahmen mit Anhang), wobei n dem Grad des Generatorpolynoms entspricht (bzw. der Anzahl der Bits des Generatorpolynoms minus eins).

Generatorpolynom: 110101 (zuvor festgelegt)
Rahmen: 11011 (Nutzdaten)
Rahmen mit Anhang: 1101100000 (das Generatorpolynom hat r Stellen, also werden r - 1 = n Nullen ergänzt; hier ist r = 6)

Nun wird der Rahmen mit Anhang von links her durch das Generatorpolynom dividiert. Dabei wird ausschließlich das exklusive OR (XOR) verwendet. Wenn man dies im ersten Schritt anwendet, wird aus 110110 XOR 110101 die Zahl 000011.

immer mit der ersten **gemeinsamen 1** anfangen

1101100000
110101
———————-
0000110000
__ 110101
———————-
____ 101 (Rest)

An den Rahmen ohne Anhang wird nun der Rest angehängt. Dieser muss ebenfalls aus n Bit (mit 0en füllen) bestehen. Damit hängen wir nun 00101 an den Rahmen an.

Übertragener Rahmen: 1101100101

Bei der Überprüfung auf Richtigkeit können folgende vier Fälle auftreten

- Der Rest der Division ist null und die Nachricht ist richtig
- Der Rest der Division ist null und die Nachricht ist fehlerhaft (dieser Fall ist unwahrscheinlich, kann aber vorkommen, wenn das Fehlerpolynom ein Vielfaches des Generatorpolynoms ist oder wenn der Fehler im Datenteil und im CRC-Wert ist)
- Der Rest der Division ist ungleich null und die Nachricht ist fehlerhaft
- Der Rest der Division ist ungleich null und die Nachricht ist richtig (dieser Fall tritt ein, wenn lediglich der angehängte Rest fehlerhaft übertragen wird; dies ist jedoch ebenfalls unwahrscheinlich, da der übertragene Rest im Vergleich zur Gesamtlänge des Pakets kurz ist)

1.23 ALOHA

ALOHA ist ein Zugriffsverfahren der Sicherungsschicht (DLL, OSI-Schicht 2) aus dem Bereich der Computernetze. Beim ALOHA kann jeder Teilnehmer zu einem beliebigen Zeitpunkt sein (stets gleich langes) Datenpaket (Rahmen, engl. frame) verschicken. Versenden mehrere Teilnehmer gleichzeitig ein Paket, so kollidieren und verstümmeln diese und müssen später neu übertragen werden. Die Teilnehmer erkennen eine Verstümmelung der Daten durch das Fehlen der Bestätigung über das Ankommen des Datenpaketes vom empfangenden Rechner. Jeder Teilnehmer muss für die erneute Übertragung eine vom Zufall bestimmte Zeitperiode warten. ALOHA ist daher wie die meisten Funkprotokolle nicht echtzeitfähig, da nicht garantiert werden kann, wann ein zu sendendes Paket tatsächlich erfolgreich übertragen wird.
Max. Durchsatz von 18 %

1.24 Slotted Aloha

Im Gegensatz zum reinen **ALOHA** darf ein Teilnehmer bei diesem Verfahren nicht zu einem beliebigen Zeitpunkt senden, sondern muss sich an fest vorgegebene Zeitscheiben (engl. slots) von der Länge eines Paketes halten. Jeder Benutzer kann jederzeit in einen dieser Slots schreiben. Wenn mehrere gleichzeitig einen Slot benutzen, kommt es zur Kollision. So können sich beim Auftreten einer Kollision die verstümmelten Pakete jeweils nur voll überdecken. Dadurch lässt sich ein **Durchsatz von 37 %** oder 1/e der maximalen Kanalkapazität erzielen, was einer Verdoppelung gegenüber dem reinen ALOHA entspricht. Synchronisiertes/Slotted ALOHA ist allerdings aufwendiger zu realisieren als das unsynchronisiertes Verfahren, da alle Teilnehmer eine einheitliche, zentrale Zeitquelle benötigen, damit die "Slots" überall synchron getaktet werden.

1.25 Chord-Algorithmus (P2P-Netzwerke)

The core usage of the **Chord protocol** is to query a key from a client (generally a node as well), i.e. to find successor(k). The basic approach is to pass the query to a node's

successor, if it cannot find the key locally. This will lead to a O(N) query time.

To avoid the linear search above, Chord implements a faster search method by requiring each node to keep a finger table containing up to m entries. The i^{th} entry of node n will contain $successor((n+2^{i-1}) mod 2^m)$. The first entry of finger table is actually the node's immediate successor (and therefore an extra successor field is not needed). Every time a node wants to look up a key k, it will pass the query to the closest successor or predecessor (depending on the finger table) of k in its finger table (the "largest" one on the circle whose ID is smaller than k), until a node finds out the key is stored in its immediate successor.

With such a finger table, the number of nodes that must be contacted to find a successor in an N-node network is $O(\log N)$.

1.26 Ausfall eines Routers

1.26.1 Mit Split Horizon

Router C versendet keine Updates von dem Netz auf das Interface von welchem er das Netz gelernt hat, und vermeidet somit Routingschleifen.

1.26.2 Poisoned Reverse

Beim **Poisoned Reverse** (mit blockierter Rückroute) werden alle über diese Schnittstelle gelernten/empfangenen Routen als „nicht erreichbar“ gekennzeichnet und zurückgesendet, indem die Anzahl der Hops direkt auf 16 gesetzt wird, so dass diese Route nicht mehr von anderen Routern berücksichtigt wird, da jeder weiterer Hop unweigerlich "unendlich" (16) bedeuten würde.

1.27 ARP

Es wird fast ausschließlich im Zusammenhang mit IPv4-Adressierung auf Ethernet-Netzen, also zur Ermittlung von MAC-Adressen zu gegebenen IP-Adressen verwendet, obwohl es nicht darauf beschränkt ist. Es wird eine ARP-Anforderung (ARP Request) mit der MAC-Adresse und der IP-Adresse des anfragenden Computers als Senderadresse und der IP-Adresse des gesuchten Computers als Empfänger-IP-Adresse an alle Computer des lokalen Netzwerkes gesendet. Als Empfänger-MAC-Adresse wird dazu die Broadcast-Adresse ff-ff-ff-ff-ff-ff16 verwendet. Empfängt ein Computer ein solches Paket, sieht er nach, ob dieses Paket seine IP-Adresse als Empfänger-IP-Adresse enthält. Wenn dies der Fall ist, antwortet er mit dem Zurücksenden seiner MAC-Adresse und IP-Adresse (ARP-Antwort oder ARP-Reply) an die MAC-Quelladresse des Anforderers. Dieser trägt nach Empfang der Antwort die empfangene Kombination von IP- und MAC-Adresse in seine ARP-Tabelle, den sogenannten ARP-Cache, ein. Für ARP-Request und ARP-Reply wird das gleiche Paket-Format verwendet.

1.28 DNS

Das Domain Name System (DNS) ist einer der wichtigsten Dienste in vielen IP-basierten Netzwerken. Seine Hauptaufgabe ist die Beantwortung von Anfragen zur Namensauflösung. Das DNS funktioniert ähnlich wie eine Telefonauskunft. Der Benutzer kennt die Domain (den für Menschen merkbaren Namen eines Rechners im Internet) – zum Beispiel example.org. Diese sendet er als Anfrage in das Internet. Die URL wird dann dort vom DNS in die zugehörige IP-Adresse (die „Anschlussnummer“ im Internet) umgewandelt – zum Beispiel eine IPv4-Adresse der Form 192.0.2.42 oder eine IPv6-Adresse wie 2001:db8:85a3:8d3:1319:8a2e:370:7347, und führt so zum richtigen Rechner.

DNS zeichnet sich aus durch

- dezentrale Verwaltung
- hierarchische Strukturierung des Namensraums in Baumform
- Eindeutigkeit der Namen
- Erweiterbarkeit

WORK IN PROGRESS!

0 Grundlagen

- R: Datenrate
- L: Paketgroesse
- D: Ausbreitungsverzoegerung
- $a = \frac{R \cdot D}{L} = \frac{l/v}{L/R}$ = Kanalpuffergroesse in Paketen

0.1 Verzoegerungen

- $d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$: Verzoegerung an einem Knoten
- d_{proc} : Verarbeitungsverzoegerung
- d_{queue} : Warteschlangenverzoegerung
- $d_{trans} = \frac{L}{R}$: Uebertragungsverzoegerung
- $d_{prop} = \frac{l}{v}$: Ausbreitungsverzoegerung

1 Transportschicht

1.1 Stop-And-Wait

1.1.1 Normierter Durchsatz (ohne Fehler)

$S = \frac{1}{1+2a}$

1.1.2 Normierter Durchsatz (mit Fehlern)

Annahmen:

- Fehler treten unabhaengig auf
- $N = \frac{1}{1-p}$: mittlere Anzahl, mit der jedes Paket gesendet werden muss
- Timeout = 2D

$S = \frac{L/R}{N \cdot (L/R+2D)} = \frac{1}{N \cdot (1+2RD/L)} = \frac{1}{N \cdot (1+2a)} = \frac{1-p}{1+2a}$

$\Rightarrow$ Sehr schlecht bei grossen a und p.

1.2 Schiebefensterprotokolle

max Fenstergroesse W, sodass keine SQN Kollision auftritt:

- Empfangsfenstergroesse: R, Sendefenstergroesse: T, $m = SQN_{max}$
- $R = 1 \Rightarrow W \overset{!}{<} m$
- $R = T = W > 1 \Rightarrow W \overset{!}{<} \frac{m+1}{2}$

1.2.1 Normierter Durchsatz (ohne Fehler)

- W: Fenstergroesse in Paketen, L: Paketlaenge

1. Fall: Fenstergroesse ausreichend ($W \geq 1 + 2a$)

$S = \frac{W \cdot L}{WL/R} \cdot \frac{1}{R} = 1$

2. Fall: Fenstergroesse zu klein ($W < 1 + 2a$)

$S = \frac{W \cdot L}{L/R+2D} \cdot \frac{1}{R} = \frac{W}{1+2a}$

1.2.2 Selective Repeat: Normierter Durchsatz (mit Fehlern)

Annahmen:

- Fehler treten unabhaengig auf
- $N = \frac{1}{1-p}$: mittlere Anzahl, mit der jedes Paket gesendet werden muss

1. Fall: Fenstergroesse ausreichend ($W \geq 1 + 2a$)

$S = \frac{1}{N} = 1 - p$

2. Fall: Fenstergroesse zu klein ($W < 1 + 2a$)

$S = \frac{W}{N \cdot (1+2a)} = \frac{W(1-p)}{1+2a}$

1.2.3 Go-Back-N: Normierter Durchsatz (mit Fehlern)

Annahmen:

- Wie bei Selective Repeat
- ausserdem: Im Fehlerfall sei das Fenster gefuellt, sodass alle Pakete des Fensters neu gesendet werden muessen.

(Anm: Herleitung im Skript ab Folie 96)

1. Fall: Fenstergroesse ausreichend ($W \geq 1 + 2a$)

$S = \frac{1-p}{1+1ap}$

2. Fall: Fenstergroesse zu klein ($W < 1 + 2a$)

$S = \frac{W(1-p)}{(1-p+Wp)\cdot(1+2a)}$

1.3 TCP

- RFCs 793, 1323, 2018, 5681, ...

2 Sicherungsschicht

2.1 Datensicherung

2.1.1 CRC

Berechnung des CRC:

- Nutzdaten D mit d Bits, Pruefdaten R mit r Bits
- Generatorpolynom G mit r+1 Bits, Bestimmt Faehigkeiten der Pruefsumme (F17)
- Berechnung: $(D, R) = D \cdot 2^r + (D \cdot 2^r \oplus G)$

2.2 Medienzugriff

2.2.1 Chipping

- "1" $\Leftrightarrow$ 1, "0" $\Leftrightarrow$ -1
- Chipping Code c mit Bits c_m
- Berechnung des Signals fuer 1 Datenbit d: $Z_{i,m} = d \cdot c_m$
- Generierung des Gesamtsignals durch Ueberlagerung (Addition) der Einzelsignale
- Rueckgewinnung des n-ten Datenbit-Stroms aus dem M-fach ueberlagerten Signal:

$d_i^n = \frac{\sum_{m=1}^{M} Z_{i,m}^* \cdot c_m^n}{M}$

2.2.2 ALOHA (Leistungsanalyse)

- N: Anzahl Knoten, p: Wahrscheinlichkeit, dass ein best. Knoten sendet
- Normierter Durchsatz (W'keit, dass bel. Knoten in bel. Slot ohne Kollision sendet): $S = N \cdot p(1-p)^{2(N-1)}$
- Sei Offered Load $G = Np$
  $\Rightarrow p = \frac{G}{N}$
  $\Rightarrow S = G(1 - \frac{G}{N})^{2(N-1)}$
  $\Rightarrow \lim\limits_{N\to\infty} = \lim\limits_{N\to\infty} G(1 - \frac{G}{N})^{2(N-1)} = G\mathrm{e}^{-2G}$
- Aus $\frac{d}{dG} G\mathrm{e}^{-2G} \stackrel{!}{=} 0$
  $\Rightarrow G = 0.5$
  $\Rightarrow S_{max} = \frac{1}{2\mathrm{e}} \approx 0.18$